

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Application Program Interface for Network Software  
Platform**

Inventor(s):  
Brad Abrams

ATTORNEY'S DOCKET NO. MS1-866US

## TECHNICAL FIELD

This invention relates to network software, such as Web applications, and to computer software development of such network software. More particularly, this invention relates to an application program interface (API) that facilitates use of a network software platform by application programs and computer hardware.

## BACKGROUND

Very early on, computer software came to be categorized as “operating system” software or “application” software. Broadly speaking, an application is software meant to perform a specific task for the computer user such as solving a mathematical equation or supporting word processing. The operating system is the software that manages and controls the computer hardware. The goal of the operating system is to make the computer resources available to the application programmer while at the same time, hiding the complexity necessary to actually control the hardware.

The operating system makes the resources available via functions that are collectively known as the Application Program Interface or API. The term API is also used in reference to a single one of these functions. The functions are often grouped in terms of what resource or service they provide to the application programmer. Application software requests resources by calling individual API functions. API functions also serve as the means by which messages and information provided by the operating system are relayed back to the application software.

In addition to changes in hardware, another factor driving the evolution of operating system software has been the desire to simplify and speed application



1 software development. Application software development can be a daunting task,  
2 sometimes requiring years of developer time to create a sophisticated program  
3 with millions of lines of code. For a popular operating system such as Microsoft  
4 Windows®, application software developers write thousands of different  
5 applications each year that utilize the operating system. A coherent and usable  
6 operating system base is required to support so many diverse application  
7 developers.

8 Often, development of application software can be made simpler by making  
9 the operating system more complex. That is, if a function may be useful to several  
10 different application programs, it may be better to write it once for inclusion in the  
11 operating system, than requiring dozens of software developers to write it dozens  
12 of times for inclusion in dozens of different applications. In this manner, if the  
13 operating system supports a wide range of common functionality required by a  
14 number of applications, significant savings in applications software development  
15 costs and time can be achieved.

16 Regardless of where the line between operating system and application  
17 software is drawn, it is clear that for a useful operating system, the API between  
18 the operating system and the computer hardware and application software is as  
19 important as efficient internal operation of the operating system itself.

20 Over the past few years, the universal adoption of the Internet, and  
21 networking technology in general, has changed the landscape for computer  
22 software developers. Traditionally, software developers focused on single-site  
23 software applications for standalone desktop computers, or LAN-based computers  
24 that were connected to a limited number of other computers via a local area  
25 network (LAN). Such software applications were typically referred to as “shrink

1 wrapped” products because the software was marketed and sold in a shrink-  
2 wrapped package. The applications utilized well-defined APIs to access the  
3 underlying operating system of the computer.

4 As the Internet evolved and gained widespread acceptance, the industry  
5 began to recognize the power of hosting applications at various sites on the World  
6 Wide Web (or simply the “Web”). In the networked world, clients from anywhere  
7 could submit requests to server-based applications hosted at diverse locations and  
8 receive responses back in fractions of a second. These Web applications, however,  
9 were typically developed using the same operating system platform that was  
10 originally developed for standalone computing machines or locally networked  
11 computers. Unfortunately, in some instances, these applications do not adequately  
12 transfer to the distributed computing regime. The underlying platform was simply  
13 not constructed with the idea of supporting limitless numbers of interconnected  
14 computers.

15 To accommodate the shift to the distributed computing environment being  
16 ushered in by the Internet, Microsoft Corporation is developing a network  
17 software platform known as the “.NET” platform (read as “Dot Net”). The  
18 platform allows developers to create Web services that will execute over the  
19 Internet. Such a dynamic shift requires a new ground-up design of an entirely new  
20 API.

21 In response to this challenge, the inventors developed a unique set of API  
22 functions for Microsoft’s .NET™ platform.  
23  
24  
25

## **SUMMARY**

An application program interface (API) provides a set of functions for application developers who build Web applications on a network platform, such as Microsoft Corporation's .NET™ platform.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™ platform, which includes an application program interface (API).

Fig. 3 is a block diagram of unique namespaces supported by the API, as well as function classes of the various API functions.

Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

## **BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC**

Accompanying this specification is a compact disc that stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98,

1 Windows® 2000, etc.). The compiled HTML help file stored on the compact disk  
2 is hereby incorporated by reference.

3 Additionally, the APIs contained in the compiled HTML help file are also  
4 provided in approximately 100 separate text files named "NamespaceName.txt".  
5 The text files comply with the ASCII format.

6 The compact disc itself is a CD-ROM, and conforms to the ISO 9660  
7 standard.

### 8 9 **DETAILED DESCRIPTION**

10 This disclosure addresses an application program interface (API) for a  
11 network platform upon which developers can build Web applications and services.  
12 More particularly, an exemplary API is described for the .NET™ platform created  
13 by Microsoft Corporation. The .NET™ platform is a software platform for Web  
14 services and Web applications implemented in the distributed computing  
15 environment. It represents the next generation of Internet computing, using open  
16 communication standards to communicate among loosely coupled Web services  
17 that are collaborating to perform a particular task.

18 In the described implementation, the .NET™ platform utilizes XML  
19 (extensible markup language), an open standard for describing data. XML is  
20 managed by the World Wide Web Consortium (W3C). XML is used for defining  
21 data elements on a Web page and business-to-business documents. XML uses a  
22 similar tag structure as HTML; however, whereas HTML defines how elements  
23 are displayed, XML defines what those elements contain. HTML uses predefined  
24 tags, but XML allows tags to be defined by the developer of the page. Thus,  
25 virtually any data items can be identified, allowing Web pages to function like

1 database records. Through the use of XML and other open protocols, such as  
2 Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of  
3 a wide range of services that can be tailored to the needs of the user. Although the  
4 embodiments described herein are described in conjunction with XML and other  
5 open standards, such are not required for the operation of the claimed invention.  
6 Other equally viable technologies will suffice to implement the inventions  
7 described herein.

8 As used herein, the phrase application program interface or API includes  
9 traditional interfaces that employ method or function calls, as well as remote calls  
10 (e.g., a proxy, stub relationship) and SOAP/XML invocations.

#### 12 EXEMPLARY NETWORK ENVIRONMENT

13 Fig. 1 shows a network environment 100 in which a network platform, such  
14 as the .NET™ platform, may be implemented. The network environment 100  
15 includes representative Web services 102(1), ..., 102(N), which provide services  
16 that can be accessed over a network 104 (e.g., Internet). The Web services,  
17 referenced generally as number 102, are programmable application components  
18 that are reusable and interact programmatically over the network 104, typically  
19 through industry standard Web protocols, such as XML, SOAP, WAP (wireless  
20 application protocol), HTTP (hypertext transport protocol), and SMTP (simple  
21 mail transfer protocol) although other means of interacting with the Web services  
22 over the network may also be used, such as Remote Procedure Call (RPC) or  
23 object broker type technology. A Web service can be self-describing and is often  
24 defined in terms of formats and ordering of messages.

1 Web services 102 are accessible directly by other services (as represented  
2 by communication link 106) or a software application, such as Web application  
3 110 (as represented by communication links 112 and 114). Each Web service 102  
4 is illustrated as including one or more servers that execute software to handle  
5 requests for particular services. Such services often maintain databases that store  
6 information to be served back to requesters. Web services may be configured to  
7 perform any one of a variety of different services. Examples of Web services  
8 include login verification, notification, database storage, stock quoting, location  
9 directories, mapping, music, electronic wallet, calendar/scheduler, telephone  
10 listings, news and information, games, ticketing, and so on. The Web services can  
11 be combined with each other and with other applications to build intelligent  
12 interactive experiences.

13 The network environment 100 also includes representative client devices  
14 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as  
15 represented by communication link 122) and/or the Web application 110 (as  
16 represented by communication links 124, 126, and 128). The clients may  
17 communicate with one another using standard protocols as well, as represented by  
18 an exemplary XML link 130 between clients 120(3) and 120(4).

19 The client devices, referenced generally as number 120, can be  
20 implemented many different ways. Examples of possible client implementations  
21 include, without limitation, portable computers, stationary computers, tablet PCs,  
22 televisions/set-top boxes, wireless communication devices, personal digital  
23 assistants, gaming consoles, printers, photocopiers, and other smart devices.

24 The Web application 110 is an application designed to run on the network  
25 platform and may utilize the Web services 102 when handling and servicing

1 requests from clients 120. The Web application 110 is composed of one or more  
2 software applications 130 that run atop a programming framework 132, which are  
3 executing on one or more servers 134 or other computer systems. Note that a  
4 portion of Web application 110 may actually reside on one or more of clients 120.  
5 Alternatively, Web application 110 may coordinate with other software on clients  
6 120 to actually accomplish its tasks.

7 The programming framework 132 is the structure that supports the  
8 applications and services developed by application developers. It permits multi-  
9 language development and seamless integration by supporting multiple languages.  
10 It supports open protocols, such as SOAP, and encapsulates the underlying  
11 operating system and object model services. The framework provides a robust and  
12 secure execution environment for the multiple programming languages and offers  
13 secure, integrated class libraries.

14 The framework 132 is a multi-tiered architecture that includes an  
15 application program interface (API) layer 142, a common language runtime (CLR)  
16 layer 144, and an operating system/services layer 146. This layered architecture  
17 allows updates and modifications to various layers without impacting other  
18 portions of the framework. A common language specification (CLS) 140 allows  
19 designers of various languages to write code that is able to access underlying  
20 library functionality. The specification 140 functions as a contract between  
21 language designers and library designers that can be used to promote language  
22 interoperability. By adhering to the CLS, libraries written in one language can be  
23 directly accessible to code modules written in other languages to achieve seamless  
24 integration between code modules written in one language and code modules  
25 written in another language. One exemplary detailed implementation of a CLS is

described in an ECMA standard created by participants in ECMA TC39/TG3.

The reader is directed to the ECMA web site at [www.ecma.ch](http://www.ecma.ch).

The API layer 142 presents groups of functions that the applications 130 can call to access the resources and services provided by layer 146. By exposing the API functions for a network platform, application developers can create Web applications for distributed computing systems that make full use of the network resources and other Web services, without needing to understand the complex interworkings of how those network resources actually operate or are made available. Moreover, the Web applications can be written in any number of programming languages, and translated into an intermediate language supported by the common language runtime 144 and included as part of the common language specification 140. . In this way, the API layer 142 can provide methods for a wide and diverse variety of applications.

Additionally, the framework 132 can be configured to support API calls placed by remote applications executing remotely from the servers 134 that host the framework. Representative applications 148(1) and 148(2) residing on clients 120(3) and 120(M), respectively, can use the API functions by making calls directly, or indirectly, to the API layer 142 over the network 104.

The framework may also be implemented at the clients. Client 120(3) represents the situation where a framework 150 is implemented at the client. This framework may be identical to server-based framework 132, or modified for client purposes. Alternatively, the client-based framework may be condensed in the event that the client is a limited or dedicated function device, such as a cellular phone, personal digital assistant, handheld computer, or other communication/computing device.



## DEVELOPERS' PROGRAMMING FRAMEWORK

Fig. 2 shows the programming framework 132 in more detail. The common language specification (CLS) layer 140 supports applications written in a variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python, and so on. The common language specification 140 specifies a subset of features or rules about features that, if followed, allow the various languages to communicate. For example, some languages do not support a given type (e.g., an "int\*" type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the "int[]" type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled "Method and System for Compiling Multiple Languages", filed 6/21/2000 (serial

number 09/598,105) and “Unified Data Type System and Method” filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security, authentication, verification, processes and threads, memory management, and so on. The object model services 146(2) provide interfacing with other objects to perform various tasks. Calls made to the API layer 142 are handed to the common language runtime layer 144 for local execution by the operating system 146(1) and/or object model services 146(2).

The API 142 groups API functions into multiple namespaces. Namespaces essentially define a collection of classes, interfaces, delegates, enumerations, and structures, which are collectively called “types”, that provide a specific set of related functionality. A class represents managed heap allocated data that has reference assignment semantics. A delegate is an object oriented function pointer. An enumeration is a special kind of value type that represents named constants. A structure represents static allocated data that has value assignment semantics. An interface defines a contract that other types can implement.

By using namespaces, a designer can organize a set of types into a hierarchical namespace. The designer is able to create multiple groups from the set of types, with each group containing at least one type that exposes logically related functionality. In the exemplary implementation, the API 142 is organized into four root namespaces: a first namespace 200 for Web applications, a second

namespace 202 for client applications, a third namespace 204 for data and XML, and a fourth namespace 206 for base class libraries (BCLs). Each group can then be assigned a name. For instance, types in the Web applications namespace 200 are assigned the name “Web”, and types in the data and XML namespace 204 can be assigned names “Data” and “XML” respectively. The named groups can be organized under a single “global root” namespace for system level APIs, such as an overall System namespace. By selecting and prefixing a top level identifier, the types in each group can be easily referenced by a hierarchical name that includes the selected top level identifier prefixed to the name of the group containing the type. For instance, types in the Web applications namespace 200 can be referenced using the hierarchical name “System.Web”. In this way, the individual namespaces 200, 202, 204, and 206 become major branches off of the System namespace and can carry a designation where the individual namespaces are prefixed with a designator, such as a “System.” prefix.

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web pages (e.g., Microsoft’s Active Server Pages (ASP)). It supplies types that enable browser/server communication. The client applications namespace 202 pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional (2D), imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

The data and XML namespace 204 relates to connectivity to data sources and XML functionality. It supplies classes, interfaces, delegates, and enumerations that enable security, specify data types, and serialize objects into XML format documents or streams. The base class libraries (BCL) namespace

206 pertains to basic system and runtime functionality. It contains the fundamental types and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

In addition to the framework 132, programming tools 210 are provided to assist the developer in building Web services and/or applications. One example of the programming tools 200 is Visual Studio™, a multi-language suite of programming tools offered by Microsoft Corporation.

#### ROOT API NAMESPACES

Fig. 3 shows the API 142 and its four root namespaces in more detail. In one embodiment, the namespaces are identified according to a hierarchical naming convention in which strings of names are concatenated with periods. For instance, the Web applications namespace 200 is identified by the root name “System.Web”. Within the “System.Web” namespace is another namespace for Web services, identified as “System.Web.Services”, which further identifies another namespace for a description known as “System.Web.Services.Description”. With this naming convention in mind, the following provides a general overview of selected namespaces of the API 142, although other naming conventions could be used with equal effect.

The Web applications namespace 200 (“System.Web”) defines additional namespaces, including:

- A services namespace 300 (“System.Web.Services”) containing classes that enable a developer to build and use Web services. The

1 services namespace 300 defines additional namespaces, including a  
2 description namespace 302 ("System.Web.Services.Description")  
3 containing classes that enable a developer to publicly describe a  
4 Web service via a service description language (such as WSDL, a  
5 specification available from the W3C), a discovery namespace 304  
6 ("System.Web.Services.Discovery") containing classes that allow  
7 Web service consumers to locate available Web Services on a Web  
8 server, and a protocols namespace 306  
9 ("System.Web.Services.Protocols") containing classes that define  
10 the protocols used to transmit data across a network during  
11 communication between Web service clients and the Web service  
12 itself.

- 13 • A caching namespace 308 ("System.Web.Caching") containing  
14 classes that enable developers to decrease Web application response  
15 time through temporarily caching frequently used resources on the  
16 server. This includes ASP.NET pages, web services, and user  
17 controls. (ASP.NET is the updated version of Microsoft's ASP  
18 technology.) Additionally, a cache dictionary is available for  
19 developers to store frequently used resources, such as hash tables  
20 and other data structures.
- 21 • A configuration namespace 310 ("System.Web.Configuration")  
22 containing classes that are used to read configuration data in for an  
23 application.
- 24 • A UI namespace 312 ("System.Web.UI") containing types that allow  
25 developers to create controls and pages that will appear in Web

1 applications as user interfaces on a Web page. This namespace  
2 includes the control class, which provides all web based controls,  
3 whether those encapsulating HTML elements, higher level Web  
4 controls, or even custom User controls, with a common set of  
5 functionality. Also provided are classes which provide the web  
6 forms server controls data binding functionality, the ability to save  
7 the view state of a given control or page, as well as parsing  
8 functionality for both programmable and literal controls. Within the  
9 UI namespace 312 are two additional namespaces: an HTML  
10 controls namespace 314 (“System.Web.UI.HtmlControls”) containing  
11 classes that permit developers to interact with types that  
12 encapsulates html 3.2 elements create HTML controls, and a Web  
13 controls namespace 316 (“System.Web.UI.WebControls”) containing  
14 classes that allow developers to create higher level Web  
15 controls.

- 16 • A security namespace 318 (“System.Web.Security”) containing  
17 classes used to implement security in web server applications, such  
18 as basic authentication, challenge response authentication, and role  
19 based authentication.
- 20 • A session state namespace 320 (“System.Web.SessionState”) containing  
21 classes used to access session state values (i.e., data that  
22 lives across requests for the lifetime of the session) as well as  
23 session-level settings and lifetime management methods.

24  
25 The client applications namespace 202 is composed of two namespaces:

- A windows forms namespace 322 (“System.Windows.Forms”) containing classes for creating Windows®-based client applications that take full advantage of the rich user interface features available in the Microsoft Windows® operating system, such as the ability to drag and drop screen elements. Such classes may include wrapped APIs available in the Microsoft Windows® operating system that are used in a windowing UI environment. Within this namespace are a design namespace 324 (“System.Windows.Forms.Design”) that contains classes to extend design-time support for Windows forms and a component model namespace 326 (“System.Windows.Forms.ComponentModel”) that contains the windows form implementation of the general component model defined in System.ComponentModel. This namespace contains designer tools, such as Visual Studio, which offer a rich experience for developers at design time.
- A drawing namespace 328 (“System.Drawing”) containing classes for graphics functionality. The drawing namespace 328 includes a 2D drawing namespace 330 (“System.Drawing.Drawing2D”) that contains classes and enumerations to provide advanced 2-dimensional and vector graphics functionality, an imaging namespace 332 (“System.Drawing.Imaging”) that contains classes for advanced imaging functionality, a printing namespace 334 (“System.Drawing.Printing”) that contains classes to permit developers to customize printing, and a text namespace 336

1 (“System.Drawing.Text”) that contains classes for advanced  
2 typography functionality.

3  
4 The data and XML namespace 204 is composed of two namespaces:

- 5  
6 • A data namespace 340 (“System.Data”) containing classes that  
7 enable developers to build components that efficiently manage data  
8 from multiple data sources. It implements an architecture that, in a  
9 disconnected scenario (such as the Internet), provides tools to  
10 request, update, and reconcile data in multiple tier systems. The data  
11 namespace 340 includes a common namespace 342 that contains  
12 types shared by data providers. A data provider describes a  
13 collection of types used to access a data source, such as a database,  
14 in the managed space. The data namespace 340 also includes an  
15 OLE DB namespace 344 that contains types pertaining to data used  
16 in object-oriented databases (e.g., Microsoft’s SQL Server), and a  
17 SQL client namespace 346 that contains types pertaining to data  
18 used by SQL clients. The data namespace also includes a SQL types  
19 namespace 348 (“System.Data.SqlTypes”) that contains classes for  
20 native data types within Microsoft’s SQL Server. The classes  
21 provide a safer, faster alternative to other data types. Using the  
22 objects within this namespace helps prevent type conversion errors  
23 caused in situations where loss of precision could occur. Because  
24 other data types are converted to and from SQL types behind the  
25



1 scenes, explicitly creating and using objects within this namespace  
2 results in faster code as well.

- 3 • An XML namespace 350 (“System.XML”) containing classes that  
4 provide standards-based support for processing XML. The supported  
5 standards include XML (e.g., version 1.0), XML Namespaces (both  
6 stream level and DOM), XML Schemas, XPath expressions, XSL/T  
7 transformations, DOM Level 2 Core, and SOAP (e.g., version 1.1).  
8 The XML namespace 350 includes an XSLT namespace 352  
9 (“System.XML.Xsl”) that contains classes and enumerations to  
10 support XSLT (Extensible Stylesheet Language Transformations),  
11 an XPath namespace 354 (“System.XML.XPath”) that contains an  
12 XPath parser and evaluation engine, and a serialization namespace  
13 356 (“System.XML.Serialization”) that contains classes used to  
14 serialize objects into XML format documents or streams.

15  
16 The base class library namespace 206 (“System”) includes the following  
17 namespaces:

- 18  
19 • A collections namespace 360 (“System.Collections”) containing  
20 interfaces and classes that define various collections of objects, such  
21 as lists, queues, arrays, hash tables and dictionaries.
- 22 • A configuration namespace 362 (“System.Configuration”)  
23 containing classes and interfaces that allow developers to  
24 programmatically access configuration settings and handle errors in  
25 configuration files.

- A diagnostics namespace 364 (“System.Diagnostics”) containing classes that are used to debug applications and to trace code execution. The namespace allows developers to start system processes, read and write to event logs, and monitor system performance using performance counters.
- A globalization namespace 366 (“System.Globalization”) containing classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers, and the sort order for strings.
- An I/O namespace 368 (“System.IO”) containing the infrastructure pieces to operate with the input/output of data streams, files, and directories. This namespace includes a model for working with streams of bytes, higher level readers and writers which consume those bytes, various constructions or implementations of the streams (e.g., FileStream and MemoryStream) and, a set of utility classes for working with files and directories.
- A net namespace 370 (“System.Net”) providing an extensive set of classes for building network-enabled application, referred to as the Net Class Libraries (NCL). One element to the design of the Net Class Libraries is an extensible, layered approach to exposing networking functionality. The NCL stack contains three basic layers. A base layer (System.Net.Socket) provides access to an interface to TCP/IP, the communications protocol of UNIX networks and the Internet. One example of such an interface is the “WinSock

API” from Microsoft Corporation. The next layer is the Transport Protocol classes, which support such transport protocols as TCP and UDP. Developers may write their own protocol classes to provide support for protocols such as IGMP and ICMP. The third layer is the Web request, which provides an abstract factory pattern for the creation of other protocol classes. The NCL provides implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace (“System.Reflection”) 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 (“System.Resources”) containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 (“System.Security”) supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 (“System.ServiceProcess”) containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.

- A text namespace 380 (“System.Text”) containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.
- A threading namespace 382 (“System.Threading”) containing classes and interfaces that enable multi-threaded programming. The threading namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually-exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.
- A runtime namespace 384 (“System.Runtime”) containing multiple namespaces concerning runtime features, including an interoperation services namespace 386 (“System.Runtime.InteropServices”) that contains a collection of classes useful for accessing COM objects. The types in the InteropServices namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class. The runtime namespace 384 further includes a remoting namespace 388 (“System.Runtime.Remoting”) that contains classes and interfaces allowing developers to create and configure distributed applications. Another namespace within the runtime namespace 384 is a

1 serialization namespace 390 (“System.Runtime.Serialization”) that  
2 contains classes used for serializing and deserializing objects.  
3 Serialization is the process of converting an object or a graph of  
4 objects into a linear sequence of bytes for either storage or  
5 transmission to another location.

6  
7 The following description pertains to the “.forms” namespace and describes  
8 various types associated with the namespace as well as the functionalities provided  
9 by the various types.

#### 10 11 1. System.Windows.Forms.Design

12 The namespace contains classes that can be used to extend design-time  
13 support for Windows Forms.

##### 14 15 *Description*

16 The **System.Windows.Forms.Design** namespace contains classes that can be  
17 used to extend design-time support for Windows Forms.

18 AnchorEditor class (System.Windows.Forms.Design)

##### 19 20 *Description*

21 Provides a user interface for configuring an  
22 **System.Windows.Forms.Control.Anchor** property.

23 The **System.Windows.Forms.Design.AnchorEditor** provides a design-time  
24 user interface for configuring an **System.Windows.Forms.Control.Anchor**  
25 property. An **System.Windows.Forms.Control.Anchor** property is typically  
used to determine which sides of the container a control is bound to. This class

provides a drop-down graphical control that allows the user to specify which sides of the container to anchor the control to.

## 1. Constructors:

### a) *AnchorEditor*

*Example Syntax:*

```
[C#] public AnchorEditor();
[C++] public: AnchorEditor();
[VB] Public Sub New()
[JavaScript] public function AnchorEditor();
```

## 2. Methods:

### a) *EditValue*

```
[C#] public override object EditValue(ITypeDescriptorContext context,
IServiceProvider provider, object value);
[C++] public: Object* EditValue(ITypeDescriptorContext* context,
IServiceProvider* provider, Object* value);
[VB] Overrides Public Function EditValue(ByVal context As
ITypeDescriptorContext, ByVal provider As IServiceProvider, ByVal value As
Object) As Object
[JavaScript] public override function EditValue(context : ITypeDescriptorContext,
provider : IServiceProvider, value : Object) : Object;
```

### *Description*

Edits the value of the specified object using the specified service provider and context.

*Return Value:* The new value of the object. If the value of the object hasn't changed, this should return the same object it was passed.

A service provider is provided so that any required editing services can be obtained. An **System.ComponentModel.ITypeDescriptorContext** object that can be used to gain additional context information. A service provider object through which editing services may be obtained. An instance of the value being edited.

*b) GetEditStyle*

```
[C#]          public          override          UITypeEditorEditStyle
GetEditStyle(ITypeDescriptorContext          context);

[C++] public:  UITypeEditorEditStyle  GetEditStyle(ITypeDescriptorContext*
context);

[VB]  Overrides  Public  Function  GetEditStyle(ByVal  context  As
ITypeDescriptorContext)          As          UITypeEditorEditStyle

[JScript] public override function GetEditStyle(context : ITypeDescriptorContext)
:          UITypeEditorEditStyle;
```

*Description*

Gets the editor style used by the **System.Windows.Forms.Design.AnchorEditor.EditValue(System.ComponentModel.ITypeDescriptorContext, System.IServiceProvider, System.Object)** method.

*Return Value:* One of the **System.Drawing.Design.UITypeEditorEditStyle** values indicating the provided editing style. If the method is not supported, this method will return **System.Drawing.Design.UITypeEditorEditStyle.None**. An **System.ComponentModel.ITypeDescriptorContext** object that can be used to gain additional context information.

AxImporter class (System.Windows.Forms.Design)

*a) ToString*

*Description*

Imports ActiveX controls and generates a wrapper that can be accessed by a designer.

*b) AxImporter*

*Example Syntax:*

*c) ToString*

[C#] public AxImporter(AxImporter.Options options);

[C++] public: AxImporter(AxImporter.Options\* options);

[VB] Public Sub New(ByVal options As AxImporter.Options)

[JScript] public function AxImporter(options : AxImporter.Options);

*Description*

**2. Properties:**

*a) GeneratedAssemblies*

*b) ToString*

[C#] public string[] GeneratedAssemblies {get;}

[C++] public: \_\_property String\* get\_GeneratedAssemblies();

[VB] Public ReadOnly Property GeneratedAssemblies As String ()



1 [JScript] public function get GeneratedAssemblies() : String[];

3 *Description*

5 c) *GeneratedSources*

6 d) *ToString*

8 [C#] public string[] GeneratedSources {get;}

9 [C++] public: \_\_property String\* get\_GeneratedSources();

10 [VB] Public ReadOnly Property GeneratedSources As String ()

11 [JScript] public function get GeneratedSources() : String[];

12 e) *GeneratedTypeLibAttributes*

13 f) *ToString*

15 [C#] public TYPELIBATTR[] GeneratedTypeLibAttributes {get;}

16 [C++] public: \_\_property TYPELIBATTR get\_GeneratedTypeLibAttributes();

17 [VB] Public ReadOnly Property GeneratedTypeLibAttributes As TYPELIBATTR  
18 ()

19 [JScript] public function get GeneratedTypeLibAttributes() : TYPELIBATTR[];

21 *Description*

g) *GenerateFromFile*

```
[C#]      public      string      GenerateFromFile(FileInfo      file);
[C++]     public:     String*      GenerateFromFile(FileInfo*      file);
[VB]      Public      Function      GenerateFromFile(ByVal file As FileInfo) As String
[JavaScript] public function GenerateFromFile(file : FileInfo) : String; Generates a
wrapper for an ActiveX control for use in the design-time environment.
```

*Description*

Generates a wrapper for an ActiveX control for use in the design-time environment. The file containing the control.

h) *GenerateFromTypeLibrary*

```
[C#]      public      string      GenerateFromTypeLibrary(UCOMITypeLib typeLib);
[C++]     public:     String*      GenerateFromTypeLibrary(UCOMITypeLib* typeLib);
[VB]      Public      Function      GenerateFromTypeLibrary(ByVal typeLib As
UCOMITypeLib)                                     As String
[JavaScript] public function GenerateFromTypeLibrary(typeLib : UCOMITypeLib) :
String; Generates a wrapper for an ActiveX control for use in the design-time
environment.
```

*Description*

Generates a wrapper for an ActiveX control for use in the design-time environment.

i) *GenerateFromTypeLibrary*

```
[C#] public string GenerateFromTypeLibrary(UCOMITypeLib typeLib, Guid
clsid);
[C++] public: String* GenerateFromTypeLibrary(UCOMITypeLib* typeLib,
Guid
clsid);
[VB] Public Function GenerateFromTypeLibrary(ByVal typeLib As
UCOMITypeLib, ByVal clsid As Guid) As String
[JScript] public function GenerateFromTypeLibrary(typeLib : UCOMITypeLib,
clsid : Guid) : String;
```

*Description*

Generates a wrapper for an ActiveX control for use in the design-time environment. A GUID for the wrapper.

j) *GetFileOfTypeLib*

```
[C#] public static string GetFileOfTypeLib(ref TYPELIBATTR tlibattr);
[C++] public: static String* GetFileOfTypeLib(TYPELIBATTR* tlibattr);
[VB] Public Shared Function GetFileOfTypeLib(ByRef tlibattr As
TYPELIBATTR) As String
[JScript] public static function GetFileOfTypeLib(tlibattr : TYPELIBATTR) :
String; Gets the file name corresponding to the given TypelibAttribute.
```

AxParameterData class (System.Windows.Forms.Design)

*a) ToString*

*Description*

Represents a parameter of a method of a hosted ActiveX control.

*b) AxParameterData*

*Example Syntax:*

*c) ToString*

[C#] public AxParameterData(ParameterInfo info);

[C++] public: AxParameterData(ParameterInfo\* info);

[VB] Public Sub New(ByVal info As ParameterInfo)

[JScript] public function AxParameterData(info : ParameterInfo);

*Description*

Initializes a new instance of the **System.Windows.Forms.Design.AxParameterData** class using the specified parameter information. A **System.Reflection.ParameterInfo** indicating the parameter information to use.

*d) AxParameterData*

*Example Syntax:*

*e) ToString*

[C#] public AxParameterData(ParameterInfo info, bool ignoreByRefs);

[C++] public: AxParameterData(ParameterInfo\* info, bool ignoreByRefs);

1 [VB] Public Sub New(ByVal info As ParameterInfo, ByVal ignoreByRefs As  
2 Boolean)

3 [JScript] public function AxParameterData(info : ParameterInfo, ignoreByRefs :  
4 Boolean);

5  
6 *Description*

7 Initializes a new instance of the  
8 **System.Windows.Forms.Design.AxParameterData** class using the  
9 specified parameter information and whether to ignore by reference parameters.  
A **System.Reflection.ParameterInfo** indicating the parameter information to  
use. A value indicating whether to ignore parameters passed by reference.

10 *f) AxParameterData*

11 *Example Syntax:*

12 *g) ToString*

13  
14 [C#] public AxParameterData(string inname, string typeName);

15 [C++] public: AxParameterData(String\* inname, String\* typeName);

16 [VB] Public Sub New(ByVal inname As String, ByVal typeName As String)

17 [JScript] public function AxParameterData(inname : String, typeName : String);

18 Initializes a new instance of the  
19 **System.Windows.Forms.Design.AxParameterData** class using the specified  
20 name and type name.

21  
22 *Description*

23 Initializes a new instance of the  
24 **System.Windows.Forms.Design.AxParameterData** class using the  
25 specified name and type name. The name of the parameter. The name of the  
type expected by the parameter.

h) *AxParameterData*

*Example Syntax:*

i) *ToString*

```
[C#]      public      AxParameterData(string      inname,      Type      type);
[C++]     public:     AxParameterData(String*      inname,      Type*      type);
[VB]      Public Sub New(ByVal inname As String, ByVal type As Type)
[JScript] public function AxParameterData(inname : String, type : Type);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.Design.AxParameterData** class using the specified name and type. The name of the parameter. The type expected by the parameter.

j) *Direction*

k) *ToString*

```
[C#]      public      FieldDirection      Direction      {get;}
[C++]     public:     __property      FieldDirection      get_Direction();
[VB]      Public      ReadOnly      Property      Direction      As      FieldDirection
[JScript] public      function      get      Direction()      :      FieldDirection;
```

*Description*

Indicates the direction of assignment fields.

l) *IsByRef*

m) *ToString*

[C#]            public            bool            IsByRef            {get;}

[C++]           public:            \_\_property           bool            get\_IsByRef();

[VB]    Public    ReadOnly    Property    IsByRef    As    Boolean

[JScript]    public    function    get    IsByRef()    :    Boolean;

*Description*

Indicates whether the parameter data is passed by reference.

n) *IsIn*

o) *ToString*

[C#]            public            bool            IsIn            {get;}

[C++]           public:            \_\_property           bool            get\_IsIn();

[VB]    Public    ReadOnly    Property    IsIn    As    Boolean

[JScript]    public    function    get    IsIn()    :    Boolean;

*Description*

Indicates whether the parameter data is in.

p) *IsOptional*

q) *ToString*

[C#]            public            bool            IsOptional            {get;}

[C++]           public:            \_\_property           bool            get\_IsOptional();

```

1 [VB]      Public      ReadOnly      Property      IsOptional      As      Boolean
2 [JScript]  public      function      get      IsOptional()      :      Boolean;

```

#### *Description*

Indicates whether the parameter data is optional.

*r) IsOut*

*s) ToString*

```

9 [C#]          public          bool          IsOut          {get;}
10 [C++]          public:          __property          bool          get_IsOut();
11 [VB]      Public      ReadOnly      Property      IsOut      As      Boolean
12 [JScript]  public      function      get      IsOut()      :      Boolean;

```

#### *Description*

Indicates whether the parameter data is out.

*t) Name*

*u) ToString*

```

19 [C#]          public          string          Name          {get;          set;}
20 [C++]  public:  __property  String*  get_Name();public:  __property  void
21  set_Name(String*);
22 [VB]      Public          Property          Name          As          String
23 [JScript] public function get Name() : String;public function set Name(String);

```



*Description*

Gets or sets the name of the parameter.

v) *ParameterType*

w) *ToString*

[C#]            public            Type            ParameterType            {get;}

[C++]           public:           \_\_property           Type\*           get\_ParameterType();

[VB]           Public           ReadOnly           Property           ParameterType           As           Type

[JScript]       public           function           get           ParameterType()           :           Type;

*Description*

Gets or sets the type expected by the parameter.

x) *TypeName*

y) *ToString*

[C#]            public            string            TypeName            {get;}

[C++]           public:           \_\_property           String\*           get\_TypeName();

[VB]           Public           ReadOnly           Property           TypeName           As           String

[JScript]       public           function           get           TypeName()           :           String;

*Description*

Gets or sets the name of the type expected by the parameter.

z) *Convert*

```
[C#] public static AxParameterData[] Convert(ParameterInfo[] infos);  
[C++] public: static AxParameterData* Convert(ParameterInfo* infos[]) [];  
[VB] Public Shared Function Convert(ByVal infos() As ParameterInfo) As  
AxParameterData()  
[JScript] public static function Convert(infos : ParameterInfo[]) :  
AxParameterData[]; Converts the specified parameter information to an  
System.Windows.Forms.Design.AxParameterData object.
```

*Description*

Converts the specified parameter information to an  
**System.Windows.Forms.Design.AxParameterData** object.

*Return Value:* An array of

**System.Windows.Forms.Design.AxParameterData** objects representing  
the specified array of **System.Reflection.ParameterInfo** objects. An array of  
**System.Reflection.ParameterInfo** objects to convert.

aa) *Convert*

```
[C#] public static AxParameterData[] Convert(ParameterInfo[] infos, bool  
ignoreByRefs);  
[C++] public: static AxParameterData* Convert(ParameterInfo* infos[], bool  
ignoreByRefs) [];  
[VB] Public Shared Function Convert(ByVal infos() As ParameterInfo, ByVal  
ignoreByRefs As Boolean) As AxParameterData()  
[JScript] public static function Convert(infos : ParameterInfo[], ignoreByRefs :  
Boolean) : AxParameterData[];
```

*Description*

Converts the specified parameter information to an **System.Windows.Forms.Design.AxParameterData** object, according to the specified value indicating whether to ignore by reference parameters.

*Return Value:* An array of

**System.Windows.Forms.Design.AxParameterData** objects representing the specified array of **System.Reflection.ParameterInfo** objects. An array of **System.Reflection.ParameterInfo** objects to convert. A value indicating whether to ignore parameters passed by reference.

AxWrapperGen class (System.Windows.Forms.Design)

*a) ToString*

*Description*

Generates a wrapper for ActiveX controls for use in the design-time environment.

*b) ToString*

[C#]	public	static	ArrayList	GeneratedSources;
[C++]	public:	static	ArrayList*	GeneratedSources;
[VB]	Public	Shared	GeneratedSources	As ArrayList
[JScript]	public	static	var GeneratedSources	: ArrayList;

*Description*

This field initializes a local ArrayList variable named GeneratedSources.

*c) AxWrapperGen*

*Example Syntax:*

**d) ToString**

```
[C#]          public          AxWrapperGen(Type          axType);
[C++]          public:          AxWrapperGen(Type*          axType);
[VB]      Public      Sub      New(ByVal      axType      As      Type)
[JScript]      public      function      AxWrapperGen(axType      :      Type);
```

*Description*

Generates an ActiveX wrapper for design-time hosting of an ActiveX control. The type of ActiveX control to generate a wrapper for.

ComponentDocumentDesigner class (System.Windows.Forms.Design)

**a) ToString**

*Description*

Provides a Windows Forms designer implementation for designing components.

**b) ComponentDocumentDesigner**

*Example Syntax:*

**c) ToString**

```
[C#]          public          ComponentDocumentDesigner();
[C++]          public:          ComponentDocumentDesigner();
[VB]          Public          Sub          New()
[JScript] public function ComponentDocumentDesigner();
```

1        **d)      *AssociatedComponents***

2        **e)      *Component***

3        **f)      *Control***

4        **g)      *ToString***

5  
6  
7        *Description*

8        Gets the control for this designer.

9        **h)      *InheritanceAttribute***

10       **i)      *Inherited***

11       **j)      *ShadowProperties***

12       **k)      *TrayAutoArrange***

13       **l)      *ToString***

14  
15  
16       *Description*

17       Indicates whether the component tray for this designer is in auto-arrange mode.

18       **m)      *TrayLargeIcon***

19       **n)      *ToString***

20  
21       [C#]        public        bool        TrayLargeIcon        {get;        set;}

22       [C++]       public:    \_\_property bool    get\_TrayLargeIcon();public:    \_\_property void  
23       set\_TrayLargeIcon(bool);

24       [VB]        Public        Property       TrayLargeIcon       As        Boolean

[JScript] public function get TrayLargeIcon() : Boolean; public function set  
TrayLargeIcon(Boolean);

#### *Description*

Indicates whether the component tray for this designer is in large icon mode.

*o) Verbs*

*p) Dispose*

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

#### *Description*

Releases the unmanaged resources used by the **System.Windows.Forms.Design.ComponentDocumentDesigner** and optionally releases the managed resources.

This method is called by the public **Dispose()** method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

*q) GetToolSupported*

[C#] protected virtual bool GetToolSupported(ToolboxItem tool);

[C++] protected: virtual bool GetToolSupported(ToolboxItem\* tool);

[VB] Overridable Protected Function GetToolSupported(ByVal tool As  
ToolboxItem) As Boolean

1 [JScript] protected function GetToolSupported(tool : ToolboxItem) : Boolean;

3 *Description*

4 Gets a value indicating whether the specified tool is supported by this designer.  
5 *Return Value:* **true** if the tool should be enabled on the toolbox; **false** if the  
6 document designer doesn't know how to use the tool.

7 If a tool is supported then it will be enabled in the toolbox when this designer  
8 regains focus. Otherwise, it will be disabled. Once a tool is marked as enabled or  
9 disabled it may not be queried again. The tool that is to be enabled on the  
10 toolbox.

9 *r) Initialize*

10 [C#] public override void Initialize(IComponent component);

11 [C++] public: void Initialize(IComponent\* component);

12 [VB] Overrides Public Sub Initialize(ByVal component As IComponent)

13 [JScript] public override function Initialize(component : IComponent);

15 *Description*

16 Initializes the designer with the specified component.

17 The designer can get the component's site and request services from it with this  
18 call. The component to associate with this designer.

19 *s) PostFilterProperties*

20  
21 [C#] protected override void PostFilterProperties(IDictionary properties);

22 [C++] protected: void PostFilterProperties(IDictionary\* properties);

23 [VB] Overrides Protected Sub PostFilterProperties(ByVal properties As  
24 IDictionary)

1 [JScript] protected override function PostFilterProperties(properties : IDictionary);

3 *Description*

4 Allows a designer to filter the set of properties the component it is designing will expose through a **System.ComponentModel.TypeDescriptor** object.

5 *Return Value:* The augmented set of properties. If the method does not modify any properties, it may just return a reference to its input parameter. If you do make a change to the properties, you must create a new array.

7 This method is called immediately after its corresponding "Post" method. If you are overriding this method you should call the base implementation after you perform your own filtering. The properties for the class of the component.

9 t) *PreFilterProperties*

11 [C#] protected override void PreFilterProperties(IDictionary properties);

12 [C++] protected: void PreFilterProperties(IDictionary\* properties);

13 [VB] Overrides Protected Sub PreFilterProperties(ByVal properties As  
14 IDictionary)

15 [JScript] protected override function PreFilterProperties(properties : IDictionary);

17 *Description*

18 Allows a designer to filter the set of properties the component it is designing will expose through a **System.ComponentModel.TypeDescriptor** object.

19 *Return Value:* The augmented set of properties. If the method does not modify any properties, it may just return a reference to its input parameter. If you do make a change to the properties, you must create a new array to return.

21 This method is called immediately before its corresponding "Post" method. If you are overriding this method you should call the base implementation before you perform your own filtering. The properties for the class of the component.



u) *IRootDesigner.GetView*

```
1
2
3 [C#] object IRootDesigner.GetView(ViewTechnology technology);
4 [C++] Object* IRootDesigner::GetView(ViewTechnology technology);
5 [VB] Function GetView(ByVal technology As ViewTechnology) As Object
6 Implements IRootDesigner.GetView
7 [JScript] function IRootDesigner.GetView(technology : ViewTechnology) :
8 Object;
```

v) *IToolboxUser.GetToolSupported*

```
9
10
11 [C#] bool IToolboxUser.GetToolSupported(ToolboxItem tool);
12 [C++] bool IToolboxUser::GetToolSupported(ToolboxItem* tool);
13 [VB] Function GetToolSupported(ByVal tool As ToolboxItem) As Boolean
14 Implements IToolboxUser.GetToolSupported
15 [JScript] function IToolboxUser.GetToolSupported(tool : ToolboxItem) : Boolean;
```

w) *IToolboxUser.ToolPicked*

```
16
17
18 [C#] void IToolboxUser.ToolPicked(ToolboxItem tool);
19 [C++] void IToolboxUser::ToolPicked(ToolboxItem* tool);
20 [VB] Sub ToolPicked(ByVal tool As ToolboxItem) Implements
21 IToolboxUser.ToolPicked
22 [JScript] function IToolboxUser.ToolPicked(tool : ToolboxItem);
```

x) *IOleDragClient.AddComponent*

```
23
24
25 [C#] bool IOleDragClient.AddComponent(IComponent component, string name,
```

```

1  bool                                                    firstAdd);
2  [C++] bool IOleDragClient::AddComponent(IComponent* component, String*
3  name,                                                    bool            firstAdd);
4  [VB] Function AddComponent(ByVal component As IComponent, ByVal name
5  As String, ByVal firstAdd As Boolean) As Boolean Implements
6  IOleDragClient.AddComponent
7  [JScript] function IOleDragClient.AddComponent(component : IComponent,
8  name : String, firstAdd : Boolean) : Boolean;
9
10
11  y)  IOleDragClient.GetControlForComponent
12
13  [C#]  Control  IOleDragClient.GetControlForComponent(object  component);
14  [C++]  Control* IOleDragClient::GetControlForComponent(Object* component);
15  [VB]  Function  GetControlForComponent(ByVal component As Object) As
16  Control      Implements      IOleDragClient.GetControlForComponent
17  [JScript] function IOleDragClient.GetControlForComponent(component : Object)
18  : Control;
19
20  z)  IOleDragClient.GetDesignerControl
21
22  [C#]          Control      IOleDragClient.GetDesignerControl();
23  [C++]          Control*      IOleDragClient::GetDesignerControl();
24  [VB]  Function  GetDesignerControl()  As  Control  Implements
25  IOleDragClient.GetDesignerControl
26  [JScript] function IOleDragClient.GetDesignerControl() : Control;

```

aa) *IoleDragClient.IsDropOk*

```
[C#].    bool    IOleDragClient.IsDropOk(IComponent    component);
```

```
[C++]    bool    IOleDragClient::IsDropOk(IComponent*    component);
```

```
[VB] Function IsDropOk(ByVal component As IComponent) As Boolean
```

Implements	IOleDragClient.IsDropOk
------------	-------------------------

[JScript] function IOleDragClient.IsDropOk(component : IComponent) : Boolean;

## ComponentTray class (System.Windows.Forms.Design)

a) *ToString*

*Description*

Provides the user interface for the component tray of the form designer that represents non-visible components at design-time.

### *b) ComponentTray*

*Example Syntax:*

### c) *ToString*

```
[C#] public ComponentTray(IDesigner mainDesigner, IServiceProvider
serviceProvider);
```

```
[C++] public: ComponentTray(IDesigner* mainDesigner, IServiceProvider*
serviceProvider);
```

```
[VB] Public Sub New(ByVal mainDesigner As IDesigner, ByVal serviceProvider
As IServiceProvider)
```

```
[JScript] public function ComponentTray(mainDesigner : IDesigner,
```

1 serviceProvider : IServiceProvider);

2  
3 *Description*

4 Initializes a new instance of the  
5 **System.Windows.Forms.Design.ComponentTray** class using the specified  
6 designer and service provider.

7 The component tray will monitor component additions and removals and display  
8 appropriate UI objects to represent the components it contains. The main  
9 designer for the current project. An **System.IServiceProvider** that can be  
10 used to obtain design-time services.

- 11 d) *AccessibilityObject*
- 12 e) *AccessibleDefaultActionDescription*
- 13 f) *AccessibleDescription*
- 14 g) *AccessibleName*
- 15 h) *AccessibleRole*
- 16 i) *AllowDrop*
- 17 j) *Anchor*
- 18 k) *AutoArrange*
- 19 l) *ToString*

20  
21 *Description*

22 Indicates whether the positions of the tray items are automatically aligned.  
23  
24  
25

- m) *AutoScroll*
- n) *AutoScrollMargin*
- o) *AutoScrollMinSize*
- p) *AutoScrollPosition*
- q) *BackColor*
- r) *BackgroundImage*
- s) *BindingContext*
- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ClientRectangle*
- aa) *ClientSize*
- bb) *CompanyName*
- cc) *ComponentCount*
- dd) *ToString*

## *Description*

Gets the number of components contained within the tray.

*ee) Container*  
*ff) ContainsFocus*  
*gg) ContextMenu*  
*hh) Controls*  
*ii) Created*  
*jj) CreateParams*  
*kk) Cursor*  
*ll) DataBindings*  
*mm) DefaultImeMode*  
*nn) DefaultSize*  
*oo) DesignMode*  
*pp) DisplayRectangle*  
*qq) Disposing*  
*rr) Dock*  
*ss) DockPadding*  
*tt) Enabled*  
*uu) Events*  
*vv) Focused*  
*ww) Font*  
*xx) FontHeight*  
*yy) ForeColor*  
*zz) Handle*  
*aaa) HasChildren*

**bbb) Height**  
**ccc) HScroll**  
**ddd) ImeMode**  
**eee) InvokeRequired**  
**fff) IsAccessible**  
**ggg) IsDisposed**  
**hhh) IsHandleCreated**  
**iii) Left**  
**jjj) Location**  
**kkk) Name**  
**lll) Parent**  
**mmm) ProductName**  
**nnn) ProductVersion**  
**ooo) RecreatingHandle**  
**ppp) Region**  
**qqq) RenderRightToLeft**  
**rrr) ResizeRedraw**  
**sss) Right**  
**ttt) RightToLeft**  
**uuu) ShowFocusCues**  
**vvv) ShowKeyboardCues**  
**www) ShowLargeIcons**  
**xxx) ToString**

*Description*

Indicates whether the tray will display large icons for the non-visible components.

*yyy) Site*

*zzz) Size*

*aaaa) TabIndex*

*bbbb) TabStop*

*cccc) Tag*

*dddd) Text*

*eeee) Top*

*ffff) TopLevelControl*

*gggg) Visible*

*hhhh) VScroll*

*iiii) Width*

*jjjj) WindowTarget*

*kkkk) AddComponent*

[C#] public virtual void AddComponent(IComponent component);

[C++] public: virtual void AddComponent(IComponent\* component);

[VB] Overridable Public Sub AddComponent(ByVal component As IComponent)

[JScript] public function AddComponent(component : IComponent); Adds a

component to the tray.



1  
2 *Description*

3 Adds a component to the tray. The component to add.

4 **III) CanCreateComponentFromTool**

5  
6 [C#] protected virtual bool CanCreateComponentFromTool(ToolboxItem tool);

7 [C++] protected: virtual bool CanCreateComponentFromTool(ToolboxItem\*  
8 tool);

9 [VB] Overridable Protected Function CanCreateComponentFromTool(ByVal tool

10 As ToolboxItem) As Boolean

11 [JScript] protected function CanCreateComponentFromTool(tool : ToolboxItem) :

12 Boolean;

13  
14 *Description*

15 Indicates whether the specified tool can be used to create a new component.

16 *Return Value:* **true** if the specified tool can be used to create a component;  
17 otherwise, **false** . The **System.Drawing.Design.ToolboxItem** to determine  
18 whether a component can be created from.

19  
20 **mmmm)CanDisplayComponent**

21 [C#] protected virtual bool CanDisplayComponent(IComponent component);

22 [C++] protected: virtual bool CanDisplayComponent(IComponent\* component);

23 [VB] Overridable Protected Function CanDisplayComponent(ByVal component

24 As IComponent) As Boolean

25 [JScript] protected function CanDisplayComponent(component : IComponent) :

Boolean;

## *Description*

Gets a value indicating whether the specified component can be displayed.

**Return Value:** **true** if the component can be displayed; otherwise, **false** .

Some types of components, such as a Timer, may not have a UI that can be viewed at design-time. These components are usually represented by icons that are shown in the component tray. The component to determine whether can be displayed.

### *nnnn) CreateComponentFromTool*

[C#]      public      void      CreateComponentFromTool(ToolboxItem      tool);

[C++]      public:      void      CreateComponentFromTool(ToolboxItem\*      tool);

[VB]      Public      Sub      CreateComponentFromTool(ByVal tool As ToolboxItem)

[JScript]      public      function      CreateComponentFromTool(tool : ToolboxItem);

## *Description*

Creates a component in the component tray from the specified **System.Drawing.Design.ToolboxItem** .

This method checks that the component has a **System.ComponentModel.DesignTimeVisibleAttribute** before attempting to create it and add it to the component tray. The ToolboxItem to create a component from.

### *oooo) DisplayError*

[C#]              protected              void              DisplayError(Exception              e);

[C++]              protected:              void              DisplayError(Exception\*              e);

[VB]      Protected      Sub      DisplayError(ByVal e As Exception)

[JScript]      protected      function      DisplayError(e : Exception);

## Description

Displays an error message to the user with information about or from the specified exception. The exception to display.

### *pppp) Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]      protected:      void      Dispose(bool      disposing);
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)
[JScript]      protected      override      function      Dispose(disposing      :      Boolean);
```

## Description

Releases the unmanaged resources used by the **System.Windows.Forms.Design.ComponentTray** and optionally releases the managed resources.

This method is called by the public **Dispose()** method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

### *qqqq) GetLocation*

```
[C#]      public      Point      GetLocation(IComponent      receiver);
[C++]      public:      Point      GetLocation(IComponent*      receiver);
[VB]      Public      Function      GetLocation(ByVal      receiver      As      IComponent)      As      Point
[JScript]      public      function      GetLocation(receiver      :      IComponent)      :      Point;
```

## Description

Gets the location of the specified component.

**Return Value:** A **System.Drawing.Point** object indicating the coordinates of

the specified component, or an empty **System.Drawing.Point** object if the specified component could not be found in the component tray. An empty **System.Drawing.Point** object has an **System.Drawing.Point.IsEmpty** property equal to true and typically **System.Drawing.Point.X** and **System.Drawing.Point.Y** properties equal to zero.

**System.Windows.Forms.Design.ComponentTray.GetLocation(System.ComponentModel.IComponent)** is an accessor method for the location extender property that is added to each non-visual component in the component tray. The IComponent to retrieve the location of.

*rrrr) GetService*

[C#]      protected      override      object      GetService(Type      serviceType);

[C++]      protected:      Object\*      GetService(Type\*      serviceType);

[VB] Overrides Protected Function GetService(ByVal serviceType As Type) As Object

[JScript] protected override function GetService(serviceType : Type) : Object;

#### *Description*

Gets the requested service type.

*Return Value:* An instance of the requested service, or **null** if the service could not be found. The type of the service to retrieve.

*ssss) OnDoubleClick*

[C#]      protected      override      void      OnDoubleClick(EventArgs      e);

[C++]      protected:      void      OnDoubleClick(EventArgs\*      e);

[VB] Overrides Protected Sub OnDoubleClick(ByVal e As EventArgs)

[JScript] protected override function OnDoubleClick(e : EventArgs);

#### *Description*

1  
2 **tttt) OnDragDrop**

3  
4 [C#] protected override void OnDragDrop(DragEventArgs de);  
5 [C++] protected: void OnDragDrop(DragEventArgs\* de);  
6 [VB] Overrides Protected Sub OnDragDrop(ByVal de As DragEventArgs)  
7 [JScript] protected override function OnDragDrop(de : DragEventArgs);  
8

9 *Description*

10 Called in response to a drag drop for OLE drag and drop. Here we drop a toolbox  
11 component on our parent control.

12 **uuuu) OnDragEnter**

13 [C#] protected override void OnDragEnter(DragEventArgs de);  
14 [C++] protected: void OnDragEnter(DragEventArgs\* de);  
15 [VB] Overrides Protected Sub OnDragEnter(ByVal de As DragEventArgs)  
16 [JScript] protected override function OnDragEnter(de : DragEventArgs);  
17

18 *Description*

19 Called in response to a drag enter for OLE drag and drop.

20 **vvvv) OnDragLeave**

21  
22 [C#] protected override void OnDragLeave(EventArgs e);  
23 [C++] protected: void OnDragLeave(EventArgs\* e);  
24 [VB] Overrides Protected Sub OnDragLeave(ByVal e As EventArgs)  
25

1 [JScript] protected override function OnDragLeave(e : EventArgs);

3 *Description*

4 Called when a drag-drop operation leaves the control designer view Called when  
5 a drag-drop operation leaves the control designer view

6 *www)OnDragOver*

7 [C#] protected override void OnDragOver(DragEventArgs de);

8 [C++] protected: void OnDragOver(DragEventArgs\* de);

9 [VB] Overrides Protected Sub OnDragOver(ByVal de As DragEventArgs)

10 [JScript] protected override function OnDragOver(de : DragEventArgs);

12 *Description*

13 Called when a drag drop object is dragged over the control designer view Called  
14 when a drag drop object is dragged over the control designer view

15 *xxxx) OnGiveFeedback*

16  
17 [C#] protected override void OnGiveFeedback(GiveFeedbackEventArgs gfevent);

18 [C++] protected: void OnGiveFeedback(GiveFeedbackEventArgs\* gfevent);

19 [VB] Overrides Protected Sub OnGiveFeedback(ByVal gfevent As  
20 GiveFeedbackEventArgs)

21 [JScript] protected override function OnGiveFeedback(gfevent :  
22 GiveFeedbackEventArgs);

24 *Description*

Inheriting classes should override this method to handle this event. Call `base.onGiveFeedback` to send this event to any registered event listeners. event

#### *yyyy) OnLayout*

[C#]   protected   override   void   OnLayout(LayoutEventArgs   levent);  
[C++]   protected:   void   OnLayout(LayoutEventArgs\*   levent);  
[VB] Overrides Protected Sub OnLayout(ByVal levent As LayoutEventArgs)  
[JScript] protected override function OnLayout(levent : LayoutEventArgs);

#### *Description*

Forces the layout of any docked or anchored child controls.

#### *zzzz) OnLostCapture*

[C#]           protected           virtual           void           OnLostCapture();  
[C++]           protected:           virtual           void           OnLostCapture();  
[VB]           Overridable           Protected           Sub           OnLostCapture()  
[JScript]           protected                   function           OnLostCapture();

#### *Description*

This is called when we lose capture. Here we get rid of any rubber band we were drawing. You should put any cleanup code in here.

#### *aaaaa) OnMouseDown*

[C#]   protected   override   void   OnMouseDown(MouseEventArgs   e);  
[C++]   protected:   void   OnMouseDown(MouseEventArgs\*   e);  
[VB] Overrides Protected Sub OnMouseDown(ByVal e As MouseEventArgs)

1 [JScript] protected override function OnMouseDown(e : MouseEventArgs);

2  
3 *Description*

4 Inheriting classes should override this method to handle this event. Call  
5 base.OnMouseDown to send this event to any registered event listeners. event

6 *bbbbbb) OnMouseMove*

7 [C#] protected override void OnMouseMove(MouseEventArgs e);

8 [C++] protected: void OnMouseMove(MouseEventArgs\* e);

9 [VB] Overrides Protected Sub OnMouseMove(ByVal e As MouseEventArgs)

10 [JScript] protected override function OnMouseMove(e : MouseEventArgs);

11  
12 *Description*

13 Inheriting classes should override this method to handle this event. Call  
14 base.OnMouseMove to send this event to any registered event listeners. event

15 *cccccc) OnMouseUp*

16  
17 [C#] protected override void OnMouseUp(MouseEventArgs e);

18 [C++] protected: void OnMouseUp(MouseEventArgs\* e);

19 [VB] Overrides Protected Sub OnMouseUp(ByVal e As MouseEventArgs)

20 [JScript] protected override function OnMouseUp(e : MouseEventArgs);

21  
22 *Description*

23 Inheriting classes should override this method to handle this event. Call  
24 base.OnMouseUp to send this event to any registered event listeners. event



**dddd) OnPaint**

[C#]      protected      override      void      OnPaint(PaintEventArgs      pe);  
[C++]      protected:      void      OnPaint(PaintEventArgs\*      pe);  
[VB]      Overrides      Protected      Sub      OnPaint(ByVal pe As PaintEventArgs)  
[JScript]      protected      override      function      OnPaint(pe : PaintEventArgs);

*Description*

**eeee) OnResize**

[C#]      protected      override      void      OnResize(EventArgs      e);  
[C++]      protected:      void      OnResize(EventArgs\*      e);  
[VB]      Overrides      Protected      Sub      OnResize(ByVal e As EventArgs)  
[JScript]      protected      override      function      OnResize(e : EventArgs);

*Description*

**ffff) OnSetCursor**

[C#]      protected      virtual      void      OnSetCursor();  
[C++]      protected:      virtual      void      OnSetCursor();  
[VB]      Overridable      Protected      Sub      OnSetCursor()  
[JScript]      protected      function      OnSetCursor();

1  
2 *Description*

3 Sets the cursor. You can override this to set your own cursor.

4 *ggggg) RemoveComponent*

5  
6 [C#] public virtual void RemoveComponent(IComponent component);

7 [C++] public: virtual void RemoveComponent(IComponent\* component);

8 [VB] Overridable Public Sub RemoveComponent(ByVal component As  
9 IComponent)

10 [JScript] public function RemoveComponent(component : IComponent);

11  
12 *Description*

13 Removes a component from the tray. The component to remove.

14 *hhhhh) SetLocation*

15  
16 [C#] public void SetLocation(IComponent receiver, Point location);

17 [C++] public: void SetLocation(IComponent\* receiver, Point location);

18 [VB] Public Sub SetLocation(ByVal receiver As IComponent, ByVal location As  
19 Point)

20 [JScript] public function SetLocation(receiver : IComponent, location : Point);

21  
22 *Description*

23 Sets the location of the specified component to the specified location. The  
24 component to set the location of. The new location for the specified component.  
25

**iiii) IExtenderProvider.CanExtend**

[C#] bool IExtenderProvider.CanExtend(object component);

[C++] bool IExtenderProvider::CanExtend(Object\* component);

[VB] Function CanExtend(ByVal component As Object) As Boolean Implements  
IExtenderProvider.CanExtend

[JScript] function IExtenderProvider.CanExtend(component : Object) : Boolean;

**jjjj) IOleDragClient.AddComponent**

[C#] bool IOleDragClient.AddComponent(IComponent component, string name,  
bool firstAdd);

[C++] bool IOleDragClient::AddComponent(IComponent\* component, String\*  
name, bool firstAdd);

[VB] Function AddComponent(ByVal component As IComponent, ByVal name  
As String, ByVal firstAdd As Boolean) As Boolean Implements  
IOleDragClient.AddComponent

[JScript] function IOleDragClient.AddComponent(component : IComponent,  
name : String, firstAdd : Boolean) : Boolean;

**kkkkk) IOleDragClient.GetControlForComponent**

[C#] Control IOleDragClient.GetControlForComponent(object component);

[C++] Control\* IOleDragClient::GetControlForComponent(Object\* component);

[VB] Function GetControlForComponent(ByVal component As Object) As  
Control Implements IOleDragClient.GetControlForComponent

```

1 [JScript] function IOleDragClient.GetControlForComponent(component : Object)
2 : Control;

```

### 4 ***IIII) IOleDragClient.GetDesignerControl***

```

5 [C#] Control IOleDragClient.GetDesignerControl();

```

```

6 [C++] Control* IOleDragClient::GetDesignerControl();

```

```

7 [VB] Function GetDesignerControl() As Control Implements
8 IOleDragClient.GetDesignerControl

```

```

9 [JScript] function IOleDragClient.GetDesignerControl() : Control;

```

### 10 ***mmmmm)IOleDragClient.IsDropOk***

```

12 [C#] bool IOleDragClient.IsDropOk(IComponent component);

```

```

13 [C++] bool IOleDragClient::IsDropOk(IComponent* component);

```

```

14 [VB] Function IsDropOk(ByVal component As IComponent) As Boolean
15 Implements IOleDragClient.IsDropOk

```

```

16 [JScript] function IOleDragClient.IsDropOk(component : IComponent) : Boolean;

```

### 17 ***nnnnn)ISelectionUIHandler.BeginDrag***

```

19 [C#] bool ISelectionUIHandler.BeginDrag(object[] components, SelectionRules
20 rules, int initialX, int initialY);

```

```

21 [C++] bool ISelectionUIHandler::BeginDrag(Object* components __gc[],
22 SelectionRules rules, int initialX, int initialY);

```

```

23 [VB] Function BeginDrag(ByVal components() As Object, ByVal rules As
24 SelectionRules, ByVal initialX As Integer, ByVal initialY As Integer) As Boolean
25

```

```

1 Implements ISelectionUIHandler.BeginDrag
2 [JScript] function ISelectionUIHandler.BeginDrag(components : Object[], rules :
3 SelectionRules, initialX : int, initialY : int) : Boolean;
4 ooooo) ISelectionUIHandler.DragMoved
5
6 [C#] void ISelectionUIHandler.DragMoved(object[] components, Rectangle
7 offset);
8 [C++] void ISelectionUIHandler::DragMoved(Object* components __gc[],
9 Rectangle offset);
10 [VB] Sub DragMoved(ByVal components() As Object, ByVal offset As
11 Rectangle) Implements ISelectionUIHandler.DragMoved
12 [JScript] function ISelectionUIHandler.DragMoved(components : Object[], offset
13 : Rectangle);
14 ppppp) ISelectionUIHandler.EndDrag
15
16 [C#] void ISelectionUIHandler.EndDrag(object[] components, bool cancel);
17 [C++] void ISelectionUIHandler::EndDrag(Object* components __gc[], bool
18 cancel);
19 [VB] Sub EndDrag(ByVal components() As Object, ByVal cancel As Boolean)
20 Implements ISelectionUIHandler.EndDrag
21 [JScript] function ISelectionUIHandler.EndDrag(components : Object[], cancel :
22 Boolean);
23
24
25

```

**qqqqq) ISelectionUIHandler.GetComponentBounds**

```
1
2
3 [C#] Rectangle ISelectionUIHandler.GetComponentBounds(object component);
4 [C++]      Rectangle      ISelectionUIHandler::GetComponentBounds(Object*
5 component);
6 [VB] Function GetComponentBounds(ByVal component As Object) As Rectangle
7 Implements                                     ISelectionUIHandler.GetComponentBounds
8 [JScript] function ISelectionUIHandler.GetComponentBounds(component :
9 Object) : Rectangle;
```

**rrrrr) ISelectionUIHandler.GetComponentRules**

```
10
11
12 [C#]      SelectionRules      ISelectionUIHandler.GetComponentRules(object
13 component);
14 [C++]      SelectionRules      ISelectionUIHandler::GetComponentRules(Object*
15 component);
16 [VB] Function GetComponentRules(ByVal component As Object) As
17 SelectionRules      Implements      ISelectionUIHandler.GetComponentRules
18 [JScript] function ISelectionUIHandler.GetComponentRules(component : Object)
19 : SelectionRules;
```

**sssss) ISelectionUIHandler.GetSelectionClipRect**

```
20
21
22 [C#] Rectangle ISelectionUIHandler.GetSelectionClipRect(object component);
23 [C++]      Rectangle      ISelectionUIHandler::GetSelectionClipRect(Object*
24 component);
25 [VB] Function GetSelectionClipRect(ByVal component As Object) As Rectangle
```

1 Implements ISelectionUIHandler.GetSelectionClipRect

2 [JScript] function ISelectionUIHandler.GetSelectionClipRect(component : Object)

3 : Rectangle;

4 *tttt) ISelectionUIHandler.OleDragDrop*

6 [C#] void ISelectionUIHandler.OleDragDrop(DragEventArgs de);

7 [C++] void ISelectionUIHandler::OleDragDrop(DragEventArgs\* de);

8 [VB] Sub OleDragDrop(ByVal de As DragEventArgs) Implements

9 ISelectionUIHandler.OleDragDrop

10 [JScript] function ISelectionUIHandler.OleDragDrop(de : DragEventArgs);

11 *uuuuu)ISelectionUIHandler.OleDragEnter*

13 [C#] void ISelectionUIHandler.OleDragEnter(DragEventArgs de);

14 [C++] void ISelectionUIHandler::OleDragEnter(DragEventArgs\* de);

15 [VB] Sub OleDragEnter(ByVal de As DragEventArgs) Implements

16 ISelectionUIHandler.OleDragEnter

17 [JScript] function ISelectionUIHandler.OleDragEnter(de : DragEventArgs);

18 *vvvvv) ISelectionUIHandler.OleDragLeave*

20 [C#] void ISelectionUIHandler.OleDragLeave();

21 [C++] void ISelectionUIHandler::OleDragLeave();

22 [VB] Sub OleDragLeave() Implements ISelectionUIHandler.OleDragLeave

23 [JScript] function ISelectionUIHandler.OleDragLeave();

*www)ISelectionUIHandler.OleDragOver*

```
1
2
3 [C#] void ISelectionUIHandler.OleDragOver(DragEventArgs de);
4 [C++] void ISelectionUIHandler::OleDragOver(DragEventArgs* de);
5 [VB] Sub OleDragOver(ByVal de As DragEventArgs) Implements
6 ISelectionUIHandler.OleDragOver
7 [JScript] function ISelectionUIHandler.OleDragOver(de : DragEventArgs);
```

*xxxxx) ISelectionUIHandler.OnSelectionDoubleClick*

```
8
9
10 [C#] void ISelectionUIHandler.OnSelectionDoubleClick(IComponent
11 component);
12 [C++] void ISelectionUIHandler::OnSelectionDoubleClick(IComponent*
13 component);
14 [VB] Sub OnSelectionDoubleClick(ByVal component As IComponent)
15 Implements ISelectionUIHandler.OnSelectionDoubleClick
16 [JScript] function ISelectionUIHandler.OnSelectionDoubleClick(component :
17 IComponent);
```

*yyyyy) ISelectionUIHandler.QueryBeginDrag*

```
18
19
20 [C#] bool ISelectionUIHandler.QueryBeginDrag(object[] components,
21 SelectionRules rules, int initialX, int initialY);
22 [C++] bool ISelectionUIHandler::QueryBeginDrag(Object* components __gc[],
23 SelectionRules rules, int initialX, int initialY);
24 [VB] Function QueryBeginDrag(ByVal components() As Object, ByVal rules As
25 SelectionRules, ByVal initialX As Integer, ByVal initialY As Integer) As Boolean
```



1 Implements ISelectionUIHandler.QueryBeginDrag

2 [JScript] function ISelectionUIHandler.QueryBeginDrag(components : Object[],  
3 rules : SelectionRules, initialX : int, initialY : int) : Boolean;

4 *zzzzz) ISelectionUIHandler.ShowContextMenu*

5  
6 [C#] void ISelectionUIHandler.ShowContextMenu(IComponent component);

7 [C++] void ISelectionUIHandler::ShowContextMenu(IComponent\* component);

8 [VB] Sub ShowContextMenu(ByVal component As IComponent) Implements  
9 ISelectionUIHandler.ShowContextMenu

10 [JScript] function ISelectionUIHandler.ShowContextMenu(component :  
11 IComponent);

12 *aaaaaa)WndProc*

13  
14 [C#] protected override void WndProc(ref Message m);

15 [C++] protected: void WndProc(Message\* m);

16 [VB] Overrides Protected Sub WndProc(ByRef m As Message)

17 [JScript] protected override function WndProc(m : Message);

18  
19 *Description*

20 Overrides the base class's WndProc method to monitor for specific messages.  
21 The message to process.

ControlDesigner class (System.Windows.Forms.Design)

*a) WndProc*

*Description*

Provides design-time support for components that extend Control.

Users wishing to extend the design-time support of their own components should inherit from **System.Windows.Forms.Design.ControlDesigner**.

*b) WndProc*

[C#]	protected	AccessibleObject	accessibilityObj;
[C++]	protected:	AccessibleObject*	accessibilityObj;
[VB]	Protected	accessibilityObj	As AccessibleObject
[JScript]	protected var	accessibilityObj	: AccessibleObject;

*c) WndProc*

[C#]	protected	static	readonly	Point	InvalidPoint;
[C++]	protected:	static		Point	InvalidPoint;
[VB]	Protected	Shared	ReadOnly	InvalidPoint	As Point
[JScript]	protected	static	var	InvalidPoint	: Point;

*Description*

*d) ControlDesigner*

*Example Syntax:*

e) *WndProc*

```
1
2
3 [C#] public ControlDesigner();
4 [C++] public: ControlDesigner();
5 [VB] Public Sub New()
6 [JScript] public function ControlDesigner();
```

f) *AccessibilityObject*

g) *WndProc*

```
9
10 [C#] public virtual AccessibleObject AccessibilityObject {get;}
11 [C++] public: __property virtual AccessibleObject* get_AccessibilityObject();
12 [VB] Overridable Public ReadOnly Property AccessibilityObject As
13 AccessibleObject
14 [JScript] public function get AccessibilityObject() : AccessibleObject;
```

*Description*

h) *AssociatedComponents*

i) *WndProc*

```
20
21 [C#] public override ICollection AssociatedComponents {get;}
22 [C++] public: __property virtual ICollection* get_AssociatedComponents();
23 [VB] Overrides Public ReadOnly Property AssociatedComponents As ICollection
24 [JScript] public function get AssociatedComponents() : ICollection;
```

1  
2 *Description*

3 Gets or sets the collection of components associated with the designer.

4       j)     *Component*

5       k)     *Control*

6       l)     *WndProc*

7  
8  
9 *Description*

10 Gets the control being designed.

11       m)     *EnableDragRect*

12       n)     *WndProc*

13  
14 [C#]       protected       virtual       bool       EnableDragRect       {get;}

15 [C++]     protected:     \_\_property     virtual     bool     get\_EnableDragRect();

16 [VB]     Overridable     Protected     ReadOnly     Property     EnableDragRect     As     Boolean

17 [JScript]     protected     function     get     EnableDragRect()     :     Boolean;

18  
19 *Description*

20 Indicates whether drag rectangles can be drawn on this designer component.

- o) *InheritanceAttribute*
- p) *Inherited*
- q) *SelectionRules*
- r) *WndProc*

#### *Description*

Gets or sets the selection rules that indicate the movement capabilities of a component.

This should be one or more flags from the **System.Windows.Forms.Design.SelectionRules** class. If no designer provides rules for a component, the component will not get any UI services.

- s) *ShadowProperties*
- t) *Verbs*
- u) *BaseWndProc*

[C#]       protected       void       BaseWndProc(ref       Message       m);

[C++]       protected:       void       BaseWndProc(Message\*       m);

[VB]       Protected       Sub       BaseWndProc(ByRef       m       As       Message)

[JScript]       protected       function       BaseWndProc(m       :       Message);

#### *Description*

Provides default processing for messages.

This method causes the message to get processed by windows, skipping the control. This is useful if you want to block this message from getting to the control, but you do not want to block it from getting to Windows itself because it causes other messages to be generated. The message to process.

v) *CanBeParentedTo*

```
1
2
3 [C#] public virtual bool CanBeParentedTo(IDesigner parentDesigner);
4 [C++] public: virtual bool CanBeParentedTo(IDesigner* parentDesigner);
5 [VB] Overridable Public Function CanBeParentedTo(ByVal parentDesigner As
6 IDesigner) As Boolean
7 [JScript] public function CanBeParentedTo(parentDesigner : IDesigner) :
8 Boolean;
```

9  
10 *Description*

Indicates whether the specified designer can be a child of the designer.

11 *Return Value:* **true** if the designer can be parented to the specified designer;  
12 otherwise, **false**.

13 This method's result typically indicates whether the specified designer's control  
14 can be a child of the designer's control. The designer to determine whether can  
be the parent of this designer.

15 w) *DefWndProc*

```
16
17 [C#] protected void DefWndProc(ref Message m);
18 [C++] protected: void DefWndProc(Message* m);
19 [VB] Protected Sub DefWndProc(ByRef m As Message)
20 [JScript] protected function DefWndProc(m : Message);
```

21  
22 *Description*

Provides default processing for messages.

23  
24 This method causes the message to get processed by the control, rather than  
the designer. The message to process.

x) *DisplayError*

```
[C#]      protected      void      DisplayError(Exception      e);
[C++]      protected:      void      DisplayError(Exception*      e);
[VB]      Protected      Sub      DisplayError(ByVal      e      As      Exception)
[JavaScript]      protected      function      DisplayError(e      :      Exception);
```

*Description*

Displays the specified exception to the user. The exception to display.

y) *Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]      protected:      void      Dispose(bool      disposing);
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)
[JavaScript]      protected      override      function      Dispose(disposing      :      Boolean);
```

*Description*

Releases the unmanaged resources used by the **System.Windows.Forms.Design.ControlDesigner** and optionally releases the managed resources.

This method is called by the public **Dispose()** method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

z) *EnableDragDrop*

```
[C#]      protected      void      EnableDragDrop(bool      value);
[C++]      protected:      void      EnableDragDrop(bool      value);
```

1 [VB] Protected Sub EnableDragDrop(ByVal value As Boolean)

2 [JScript] protected function EnableDragDrop(value : Boolean);

4 *Description*

5 Enables or disables drag/drop support. **true** to enable drag/drop support for the  
6 control; **false** if the control should not have drag/drop support.

7 *aa) GetHitTest*

8 [C#] protected virtual bool GetHitTest(Point point);

9 [C++] protected: virtual bool GetHitTest(Point point);

10 [VB] Overridable Protected Function GetHitTest(ByVal point As Point) As  
11 Boolean

12 [JScript] protected function GetHitTest(point : Point) : Boolean;

14 *Description*

15 Indicates whether the specified point is within the bounds of the component.

16 *Return Value:* **true** if the specified point is to be handled by the component;  
17 **false** if it is not something the user and this component can interact with.

18 **System.Windows.Forms.Design.ControlDesigner.GetHitTest(System.Drawing.Point)** determines whether a specified point is within the area occupied  
19 by the control that this designer is designing. This method allows your  
20 component to support a design-time user interface that responds to clicks on its  
21 control. This method should return **true** if the specified point was within the  
22 bounds of the component, or otherwise return **false** . A point relative to the  
23 upper left of the control.

22 *bb) HookChildControls*

24 [C#] protected void HookChildControls(Control firstChild);

25 [C++] protected: void HookChildControls(Control\* firstChild);



1 [VB] Protected Sub HookChildControls(ByVal firstChild As Control)

2 [JScript] protected function HookChildControls(firstChild : Control);

3  
4 *Description*

5 Hooks the children of the specified control.

6 We need to do this for child controls that are not in design mode, which is the  
7 case for composite controls. The first child control to process. This method may  
8 recursively call itself for children of this control.

8 *cc) Initialize*

9  
10 [C#] public override void Initialize(IComponent component);

11 [C++] public: void Initialize(IComponent\* component);

12 [VB] Overrides Public Sub Initialize(ByVal component As IComponent)

13 [JScript] public override function Initialize(component : IComponent);

14  
15 *Description*

16 Initializes the designer with the specified component.

17 This method is called by the designer host to initialize the designer. The  
18 component to associate this designer with. This must always be an instance of  
19 Control.

19 *dd) InitializeNonDefault*

20  
21 [C#] public override void InitializeNonDefault();

22 [C++] public: void InitializeNonDefault();

23 [VB] Overrides Public Sub InitializeNonDefault()

24 [JScript] public override function InitializeNonDefault();

Initializes properties of the control to any non-default values.

[C#]	protected	virtual	void	OnContextMenu(int	x,	int	y);
[C++]	protected:	virtual	void	OnContextMenu(int	x,	int	y);
[VB]	Overridable Protected Sub	OnContextMenu(ByVal x As Integer, ByVal y					
	As					Integer)	
[JScript]	protected	function	OnContextMenu(x	:	int,	y	:
							int);

Called when the context menu should be displayed Called when the context menu should be displayed

[C#]	protected	virtual	void	OnCreateHandle();
[C++]	protected:	virtual	void	OnCreateHandle();
[VB]	Overridable	Protected	Sub	OnCreateHandle()
[JScript]	protected	function		OnCreateHandle();

This is called immediately after the control handle has been created.

**gg) OnDragDrop**

```
[C#]    protected    virtual    void    OnDragDrop(DragEventArgs    de);
[C++]    protected:    virtual    void    OnDragDrop(DragEventArgs*    de);
[VB]    Overridable Protected Sub OnDragDrop(ByVal de As DragEventArgs)
[JScript]    protected    function    OnDragDrop(de    :    DragEventArgs);
```

*Description*

Called when a drag drop object is dropped onto the control designer view Called when a drag drop object is dropped onto the control designer view

**hh) OnDragEnter**

```
[C#]    protected    virtual    void    OnDragEnter(DragEventArgs    de);
[C++]    protected:    virtual    void    OnDragEnter(DragEventArgs*    de);
[VB]    Overridable Protected Sub OnDragEnter(ByVal de As DragEventArgs)
[JScript]    protected    function    OnDragEnter(de    :    DragEventArgs);
```

*Description*

Called when a drag-drop operation enters the control designer view Called when a drag-drop operation enters the control designer view

**ii) OnDragLeave**

```
[C#]    protected    virtual    void    OnDragLeave(EventArgs    e);
[C++]    protected:    virtual    void    OnDragLeave(EventArgs*    e);
[VB]    Overridable Protected Sub OnDragLeave(ByVal e As EventArgs)
[JScript]    protected    function    OnDragLeave(e    :    EventArgs);
```

## *Description*

Called when a drag-drop operation leaves the control designer view Called when a drag-drop operation leaves the control designer view

### *jj) OnDragOver*

[C#] protected virtual void OnDragOver(DragEventArgs de);

[C++] protected: virtual void OnDragOver(DragEventArgs\* de);

[VB] Overridable Protected Sub OnDragOver(ByVal de As DragEventArgs)

[JScript] protected function OnDragOver(de : DragEventArgs);

## *Description*

Called when a drag drop object is dragged over the control designer view Called when a drag drop object is dragged over the control designer view

### *kk) OnGiveFeedback*

[C#] protected virtual void OnGiveFeedback(GiveFeedbackEventArgs e);

[C++] protected: virtual void OnGiveFeedback(GiveFeedbackEventArgs\* e);

[VB] Overridable Protected Sub OnGiveFeedback(ByVal e As GiveFeedbackEventArgs)

[JScript] protected function OnGiveFeedback(e : GiveFeedbackEventArgs);

## *Description*

Event handler for our GiveFeedback event, which is called when a drag operation is in progress. The host will call us with this when an OLE drag event happens.

## ll) *OnMouseDownBegin*

[C#] protected virtual void OnMouseDownBegin(int x, int y);

[C++] protected: virtual void OnMouseDownBegin(int x, int y);

[VB] Overridable Protected Sub OnMouseDownBegin(ByVal x As Integer, ByVal y As Integer)

[JScript] protected function OnMouseDownBegin(x : int, y : int);

### *Description*

Called in response to the left mouse button being pressed on a component. This does two things. First, it ensures that the component is selected. Second, it uses the ISelectionUIService to begin a drag of all selected components. The location of the mouse. The location of the mouse.

## mm) *OnMouseDownEnd*

[C#] protected virtual void OnMouseDownEnd(bool cancel);

[C++] protected: virtual void OnMouseDownEnd(bool cancel);

[VB] Overridable Protected .Sub OnMouseDownEnd(ByVal cancel As Boolean)

[JScript] protected function OnMouseDownEnd(cancel : Boolean);

### *Description*

Called at the end of a drag operation. This either commits or rolls back the drag. Set this to true to cancel the drag, or false to commit it.

## nn) *OnMouseDownMove*

[C#] protected virtual void OnMouseDownMove(int x, int y);

[C++] protected: virtual void OnMouseDownMove(int x, int y);

[VB] Overridable Protected Sub OnMouseDownMove(ByVal x As Integer, ByVal y As Integer)

[JScript] protected function OnMouseDownMove(x : int, y : int);

#### *Description*

Called for each movement of the mouse. This will check to see if a drag operation is in progress. If so, it will pass the updated drag dimensions on to the selection UI service. The x position, in screen coordinates, of the mouse. The y position, in screen coordinates, of the mouse.

#### *oo) OnMouseEnter*

[C#] protected virtual void OnMouseEnter();

[C++] protected: virtual void OnMouseEnter();

[VB] Overridable Protected Sub OnMouseEnter()

[JScript] protected function OnMouseEnter();

#### *Description*

Called when the mouse first enters the control. This is forwarded to the parent designer to enable the container selector.

#### *pp) OnMouseHover*

[C#] protected virtual void OnMouseHover();

[C++] protected: virtual void OnMouseHover();

[VB] Overridable Protected Sub OnMouseHover()

[JScript] protected function OnMouseHover();

## Description

Called after the mouse hovers over the control. This is forwarded to the parent designer to enable the container selector.

### qq) *OnMouseLeave*

[C#]	protected	virtual	void	OnMouseLeave();
[C++]	protected:	virtual	void	OnMouseLeave();
[VB]	Overridable	Protected	Sub	OnMouseLeave()
[JScript]	protected	function		OnMouseLeave();

## Description

Called when the mouse first enters the control. This is forwarded to the parent designer to enable the container selector.

### rr) *OnPaintAdornments*

[C#]	protected	virtual	void	OnPaintAdornments(PaintEventArgs pe);
[C++]	protected:	virtual	void	OnPaintAdornments(PaintEventArgs* pe);
[VB]	Overridable	Protected	Sub	OnPaintAdornments(ByVal pe As PaintEventArgs)
[JScript]	protected	function		OnPaintAdornments(pe : PaintEventArgs);

## Description

Called when the control we're designing has finished painting. This method gives the designer a chance to paint any additional adornments on top of the control. A paint event the designer may use to draw onto the control.

## ss) *OnSetComponentDefaults*

[C#]	public	override	void	OnSetComponentDefaults();
[C++]	public:		void	OnSetComponentDefaults();
[VB]	Overrides	Public	Sub	OnSetComponentDefaults()
[JScript]	public	override	function	OnSetComponentDefaults();

### *Description*

Called when the designer is initialized. This allows the designer to provide some meaningful default values in the component. The default implementation of this sets the components's default property to it's name, if that property is a string.

## tt) *OnSetCursor*

[C#]	protected	virtual	void	OnSetCursor();
[C++]	protected:	virtual	void	OnSetCursor();
[VB]	Overridable	Protected	Sub	OnSetCursor()
[JScript]	protected		function	OnSetCursor();

### *Description*

Called each time the cursor needs to be set. The ControlDesigner behavior here will set the cursor to one of three things: 1. If the toolbox service has a tool selected, it will allow the toolbox service to set the cursor. 2. If the selection UI service shows a locked selection, or if there is no location property on the control, then the default arrow will be set. 3. Otherwise, the four headed arrow will be set to indicate that the component can be clicked and moved. 4. If the user is currently dragging a component, the crosshair cursor will be used instead of the four headed arrow.



### uu) *PreFilterProperties*

```
[C#] protected override void PreFilterProperties(IDictionary properties);  
[C++] protected: void PreFilterProperties(IDictionary* properties);  
[VB] Overrides Protected Sub PreFilterProperties(ByVal properties As  
IDictionary)  
[JScript] protected override function PreFilterProperties(properties : IDictionary);
```

#### *Description*

Allows a designer to filter the set of properties the component it is designing will expose through a TypeDescriptor object.

**Return Value:** The augmented set of properties. If the method does not modify any properties, it may just return a reference to its input parameter. If you do make a change to the properties, you must create a new array.

This method is called immediately before its corresponding "Post" method. If you are overriding this method you should call the base implementation before you perform your own filtering. The properties for the class of the component.

### vv) *UnhookChildControls*

```
[C#] protected void UnhookChildControls(Control firstChild);  
[C++] protected: void UnhookChildControls(Control* firstChild);  
[VB] Protected Sub UnhookChildControls(ByVal firstChild As Control)  
[JScript] protected function UnhookChildControls(firstChild : Control);
```

#### *Description*

Hooks the children of the specified control.

We need to do this for child controls that are not in design mode, which is the case for composite controls. The first child control to process. This method may recursively call itself for children of this control.

ww) *WndProc*

```
[C#]    protected    virtual    void    WndProc(ref    Message    m);
[C++]    protected:    virtual    void    WndProc(Message*    m);
[VB]    Overridable    Protected    Sub    WndProc(ByRef    m    As    Message)
[JScript]    protected    function    WndProc(m    :    Message);
```

*Description*

This method should be called by the extending designer for each message the control would normally receive. This allows the designer to pre-process messages before allowing them to be routed to the control. The message.

ControlDesigner.ControlDesignerAccessibleObject class  
(System.Windows.Forms.Design)

a) *WndProc*

*Description*

b) *ControlDesigner.ControlDesignerAccessibleObject*

*Example Syntax:*

c) *WndProc*

```
[C#]    public    ControlDesigner.ControlDesignerAccessibleObject(ControlDesigner
designer,                                Control                                control);
[C++]    public:    ControlDesignerAccessibleObject(ControlDesigner*    designer,
Control*                                control);
```

1 [VB] Public Sub New(ByVal designer As ControlDesigner, ByVal control As  
2 Control)

3 [JScript] public function  
4 ControlDesigner.ControlDesignerAccessibleObject(designer : ControlDesigner,  
5 control : Control);

6  
7 *Description*

8  
9 d) *Bounds*

10 e) *WndProc*

11  
12 [C#] public override Rectangle Bounds {get;}

13 [C++] public: \_\_property virtual Rectangle get\_Bounds();

14 [VB] Overrides Public ReadOnly Property Bounds As Rectangle

15 [JScript] public function get Bounds() : Rectangle;

16  
17 *Description*

18  
19 f) *DefaultAction*

20 g) *WndProc*

21  
22 [C#] public override string DefaultAction {get;}

23 [C++] public: \_\_property virtual String\* get\_DefaultAction();

24 [VB] Overrides Public ReadOnly Property DefaultAction As String

1 [JScript] public function get DefaultAction() : String;

3 *Description*

5 *h) Description*

6 *i) WndProc*

8 [C#] public override string Description {get;}

9 [C++] public: \_\_property virtual String\* get\_Description();

10 [VB] Overrides Public ReadOnly Property Description As String

11 [JScript] public function get Description() : String;

13 *Description*

15 *j) Help*

16 *k) KeyboardShortcut*

17 *l) Name*

18 *m) WndProc*

21 *Description*

1        **n)     *Parent***

2        **o)     *WndProc***

3  
4    [C#]        public        override        AccessibleObject        Parent        {get;}

5    [C++]        public:        \_\_property        virtual        AccessibleObject\*        get\_Parent();

6    [VB]    Overrides    Public    ReadOnly    Property    Parent    As    AccessibleObject

7    [JScript]    public        function        get        Parent()        :        AccessibleObject;

8  
9    *Description*

10  
11        **p)     *Role***

12        **q)     *WndProc***

13  
14    [C#]        public        override        AccessibleRole        Role        {get;}

15    [C++]        public:        \_\_property        virtual        AccessibleRole        get\_Role();

16    [VB]    Overrides    Public    ReadOnly    Property    Role    As    AccessibleRole

17    [JScript]    public        function        get        Role()        :        AccessibleRole;

18  
19    *Description*

20  
21        **r)     *State***

22        **s)     *WndProc***

23  
24    [C#]        public        override        AccessibleStates        State        {get;}

25    [C++]        public:        \_\_property        virtual        AccessibleStates        get\_State();

```

1 [VB] Overrides Public ReadOnly Property State As AccessibleStates
2 [JScript] public function get State() : AccessibleStates;

```

#### *Description*

*t) Value*

*u) WndProc*

```

9 [C#] public override string Value {get;}
10 [C++] public: __property virtual String* get_Value();
11 [VB] Overrides Public ReadOnly Property Value As String
12 [JScript] public function get Value() : String;

```

#### *Description*

*v) GetChild*

```

18 [C#] public override AccessibleObject GetChild(int index);
19 [C++] public: AccessibleObject* GetChild(int index);
20 [VB] Overrides Public Function GetChild(ByVal index As Integer) As
21 AccessibleObject
22 [JScript] public override function GetChild(index : int) : AccessibleObject;

```

#### *Description*

w) ***GetChildCount***

```
[C#]      public      override      int      GetChildCount();
[C++]      public:      int      GetChildCount();
[VB]      Overrides      Public      Function      GetChildCount()      As      Integer
[JScript]      public      override      function      GetChildCount()      :      int;
```

*Description*

x) ***GetFocused***

```
[C#]      public      override      AccessibleObject      GetFocused();
[C++]      public:      AccessibleObject*      GetFocused();
[VB]      Overrides      Public      Function      GetFocused()      As      AccessibleObject
[JScript]      public      override      function      GetFocused()      :      AccessibleObject;
```

*Description*

y) ***GetSelected***

```
[C#]      public      override      AccessibleObject      GetSelected();
[C++]      public:      AccessibleObject*      GetSelected();
[VB]      Overrides      Public      Function      GetSelected()      As      AccessibleObject
[JScript]      public      override      function      GetSelected()      :      AccessibleObject;
```

*Description*

**z) *HitTest***

[C#] public override AccessibleObject HitTest(int x, int y);

[C++] public: AccessibleObject\* HitTest(int x, int y);

[VB] Overrides Public Function HitTest(ByVal x As Integer, ByVal y As Integer)

As AccessibleObject

[JScript] public override function HitTest(x : int, y : int) : AccessibleObject;

*Description*

DockEditor class (System.Windows.Forms.Design)

**a) *UseStdAccessibleObjects***

*Description*

Provides a user interface for specifying a  
**System.Windows.Forms.Control.Dock** property.

**b) *DockEditor***

*Example Syntax:*

**c) *UseStdAccessibleObjects***

[C#] public DockEditor();



1 [C++] public: DockEditor();

2 [VB] Public Sub New()

3 [JScript] public function DockEditor();

4 *d) EditValue*

6 [C#] public override object EditValue(ITypeDescriptorContext context,  
7 IServiceProvider provider, object value);

8 [C++] public: Object\* EditValue(ITypeDescriptorContext\* context,  
9 IServiceProvider\* provider, Object\* value);

10 [VB] Overrides Public Function EditValue(ByVal context As  
11 ITypeDescriptorContext, ByVal provider As IServiceProvider, ByVal value As  
12 Object) As Object

13 [JScript] public override function EditValue(context : ITypeDescriptorContext,  
14 provider : IServiceProvider, value : Object) : Object;

16 *Description*

17 Edits the specified object value using the editor style provided by  
18 GetEditorStyle. A service provider is provided so that any required editing  
19 services can be obtained.

*Return Value:* The new value of the object. If the value of the object hasn't  
19 changed, this should return the same object it was passed. An

20 **System.ComponentModel.ITypeDescriptorContext** that can be used to  
21 gain additional context information. A service provider object through which  
22 editing services may be obtained. An instance of the value being edited.

22 *e) GetEditStyle*

24 [C#] public override UITypeEditorEditStyle

25 GetEditStyle(ITypeDescriptorContext context);

```

1 [C++] public: UITypeEditorEditStyle GetEditStyle(ITypeDescriptorContext*
2 context);
3 [VB] Overrides Public Function GetEditStyle(ByVal context As
4 ITypeDescriptorContext) As UITypeEditorEditStyle
5 [JScript] public override function GetEditStyle(context : ITypeDescriptorContext)
6 : UITypeEditorEditStyle;
7

```

### *Description*

Retrieves the editing style of the EditValue method. If the method is not supported, this will return None.

*Return Value:* An enum value indicating the provided editing style. An **System.ComponentModel.ITypeDescriptorContext** that can be used to gain additional context information.

DocumentDesigner class (System.Windows.Forms.Design)

#### *a) ToString*

### *Description*

Provides a designer that extends the ScrollableControlDesigner and implements IRootDesigner.

This designer is a "root" designer meaning that it represents the document the user is working on. Any functionality that is shared between all top level documents should appear here, while things specific to only forms or user controls should go in their respective designers.

#### *b) ToString*

#### *c) DocumentDesigner*

*Example Syntax:*

- d) *ToString*
- e) *AccessibilityObject*
- f) *AssociatedComponents*
- g) *Component*
- h) *Control*
- i) *DefaultControlLocation*
- j) *DrawGrid*
- k) *EnableDragRect*
- l) *GridSize*
- m) *InheritanceAttribute*
- n) *Inherited*
- o) *SelectionRules*
- p) *ToString*

## *Description*

We override our selection rules to make the document non-sizeable.

- q) *ShadowProperties*
- r) *Verbs*
- s) *Dispose*

[C#]      protected      override      void      Dispose(bool      disposing);

[C++]      protected:      void      Dispose(bool      disposing);

1 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

2 [JScript] protected override function Dispose(disposing : Boolean);

3  
4 *Description*

5 Disposes of this designer.

6 *t) EnsureMenuEditorService*

7  
8 [C#] protected virtual void EnsureMenuEditorService(IComponent c);

9 [C++] protected: virtual void EnsureMenuEditorService(IComponent\* c);

10 [VB] Overridable Protected Sub EnsureMenuEditorService(ByVal c As  
11 IComponent)

12 [JScript] protected function EnsureMenuEditorService(c : IComponent);

13  
14 *Description*

15 Determines if a MenuEditorService has already been started. If not, this method  
16 will create a new instance of the service.

17 *u) GetToolSupported*

18 [C#] protected virtual bool GetToolSupported(ToolboxItem tool);

19 [C++] protected: virtual bool GetToolSupported(ToolboxItem\* tool);

20 [VB] Overridable Protected Function GetToolSupported(ByVal tool As  
21 ToolboxItem) As Boolean

22 [JScript] protected function GetToolSupported(tool : ToolboxItem) : Boolean;

23  
24 *Description*

Determines if the given tool is supported by this designer. If a tool is supported then it will be enabled in the toolbox when this designer regains focus. Otherwise, it will be disabled. Once a tool is marked as enabled or disabled it may not be queried again.

*Return Value:* true if the tool should be enabled on the toolbox or false if the document designer doesn't know how to use the tool. The tool that is to be enabled on the toolbox.

#### v) *Initialize*

[C#] public override void Initialize(IComponent component);

[C++] public: void Initialize(IComponent\* component);

[VB] Overrides Public Sub Initialize(ByVal component As IComponent)

[JScript] public override function Initialize(component : IComponent);

#### *Description*

Initializes the designer with the given component. The designer can get the component's site and request services from it in this call. The component to associate with this designer.

#### w) *OnContextMenu*

[C#] protected override void OnContextMenu(int x, int y);

[C++] protected: void OnContextMenu(int x, int y);

[VB] Overrides Protected Sub OnContextMenu(ByVal x As Integer, ByVal y As Integer)

[JScript] protected override function OnContextMenu(x : int, y : int);

#### *Description*

Called when the context menu should be displayed. This displays the document context menu.

x) *OnCreateHandle*

[C#]	protected	override	void	OnCreateHandle();
[C++]	protected:		void	OnCreateHandle();
[VB]	Overrides	Protected	Sub	OnCreateHandle()
[JScript]	protected	override	function	OnCreateHandle();

*Description*

This is called immediately after the control handle has been created.

y) *PreFilterProperties*

[C#]	protected	override	void	PreFilterProperties(IDictionary properties);
[C++]	protected:		void	PreFilterProperties(IDictionary* properties);
[VB]	Overrides	Protected	Sub	PreFilterProperties(ByVal properties As IDictionary)
[JScript]	protected	override	function	PreFilterProperties(properties : IDictionary);

*Description*

Allows a designer to filter the set of properties the component it is designing will expose through the TypeDescriptor object. This method is called immediately before its corresponding "Post" method. If you are overriding this method you should call the base implementation before you perform your own filtering.

*Return Value:* The augmented set of properties. If the method does not modify any properties, it may just return a reference to its input parameter. If you do make a change to the properties, you must create a new array. The properties for the class of the component.

z) *IRootDesigner.GetView*

```
[C#]    object    IRootDesigner.GetView(ViewTechnology    technology);  
[C++]   Object*    IRootDesigner::GetView(ViewTechnology    technology);  
[VB]    Function GetView(ByVal technology As ViewTechnology) As Object  
Implements                                     IRootDesigner.GetView  
[JScript] function IRootDesigner.GetView(technology : ViewTechnology) :  
Object;
```

aa) *IToolboxUser.GetToolSupported*

```
[C#]      bool      IToolboxUser.GetToolSupported(ToolboxItem      tool);  
[C++]     bool      IToolboxUser::GetToolSupported(ToolboxItem*      tool);  
[VB]      Function GetToolSupported(ByVal tool As ToolboxItem) As Boolean  
Implements                                     IToolboxUser.GetToolSupported  
[JScript] function IToolboxUser.GetToolSupported(tool : ToolboxItem) : Boolean;
```

bb) *IToolboxUser.ToolPicked*

```
[C#]      void      IToolboxUser.ToolPicked(ToolboxItem      tool);  
[C++]     void      IToolboxUser::ToolPicked(ToolboxItem*      tool);  
[VB]      Sub      ToolPicked(ByVal tool As ToolboxItem) Implements  
IToolboxUser.ToolPicked  
[JScript] function IToolboxUser.ToolPicked(tool : ToolboxItem);
```

cc) *IOleDragClient.GetControlForComponent*

```
[C#]      Control    IOleDragClient.GetControlForComponent(object    component);
```

```

1 [C++] Control* IOleDragClient::GetControlForComponent(Object* component);
2 [VB] Function GetControlForComponent(ByVal component As Object) As
3 Control Implements IOleDragClient.GetControlForComponent
4 [JScript] function IOleDragClient.GetControlForComponent(component : Object)
5 : Control;

```

#### dd) *ToolPicked*

```

8 [C#] protected virtual void ToolPicked(ToolboxItem tool);
9 [C++] protected: virtual void ToolPicked(ToolboxItem* tool);
10 [VB] Overridable Protected Sub ToolPicked(ByVal tool As ToolboxItem)
11 [JScript] protected function ToolPicked(tool : ToolboxItem);

```

#### *Description*

This will be called when the user double-clicks on a toolbox item. The document designer should create a component for the given tool. The tool to create.

#### ee) *WndProc*

```

17 [C#] protected override void WndProc(ref Message m);
18 [C++] protected: void WndProc(Message* m);
19 [VB] Overrides Protected Sub WndProc(ByRef m As Message)
20 [JScript] protected override function WndProc(m : Message);

```

#### *Description*

Overrides our base class WndProc to provide support for the menu editor service. The message to handle.



EventHandlerService class (System.Windows.Forms.Design)

**a) WndProc**

*Description*

The event handler service provides a unified way to handle the various events that our form designer must process. What we want to be able to do is to write code in one place that handles events of a certain type. We also may need to globally change the behavior of these events for modal functions like the tab order UI. Our designer, however, is in many pieces so we must somehow funnel these events to a common place. This service implements an "event stack" that contains the current set of event handlers. There can be different types of handlers on the stack. For example, we may push a keyboard handler and a mouse handler. When you request a handler, we will find the topmost handler on the stack that fits the class you requested. This way the service can be extended to any eventing scheme, and it also allows sections of a handler to be replaced (eg, you can replace mouse handling without effecting menus or the keyboard).

**b) EventHandlerService**

*Example Syntax:*

**c) WndProc**

[C#]	public	EventHandlerService(Control	focusWnd);
[C++]	public:	EventHandlerService(Control*	focusWnd);
[VB]	Public	Sub	New(ByVal focusWnd As Control)
[JScript]	public	function	EventHandlerService(focusWnd : Control);

*Description*

d) *FocusWindow*

e) *WndProc*

```
[C#]      public      Control      FocusWindow      {get;}
[C++]      public:      __property      Control*      get_FocusWindow();
[VB]      Public      ReadOnly      Property      FocusWindow      As      Control
[JavaScript]      public      function      get      FocusWindow()      :      Control;
```

*Description*

f) *WndProc*

```
[C#]      public      event      EventHandler      EventHandlerChanged;
[C++]      public:      __sealed      __event      EventHandler*      EventHandlerChanged;
[VB]      NotOverridable      Public      Event      EventHandlerChanged      As      EventHandler
```

*Description*

g) *GetHandler*

```
[C#]      public      object      GetHandler(Type      handlerType);
[C++]      public:      __sealed      Object*      GetHandler(Type*      handlerType);
[VB]      NotOverridable      Public      Function      GetHandler(ByVal handlerType As Type)
As
Object
[JavaScript]      public      function      GetHandler(handlerType : Type) : Object;
```

## Description

Gets the currently active event handler of the specified type.

**Return Value:** An instance of the handler, or null if there is no handler of the requested type.

You should never cache this value because other components are free to change it. This call is fairly quick, however, especially for cases when you are always asking for the same type of handler. The type of the handler to get.

### *h) PopHandler*

[C#]            public            void            PopHandler(object            handler);

[C++]        public:        \_\_sealed        void        PopHandler(Object\*        handler);

[VB] NotOverridable Public Sub PopHandler(ByVal handler As Object)

[JScript]    public        function        PopHandler(handler        :        Object);

## Description

Pops the given handler off of the stack.

If the handler is not the topmost component on the stack, this will pop all handlers up to and including handler. The handler to remove from the stack.

### *i) PushHandler*

[C#]            public            void            PushHandler(object            handler);

[C++]        public:        \_\_sealed        void        PushHandler(Object\*        handler);

[VB] NotOverridable Public Sub PushHandler(ByVal handler As Object)

[JScript]    public        function        PushHandler(handler        :        Object);

## Description

Pushes a new event handler on the stack.

This handler will be used by all components that request a compatible handler type. If this handler does not support the requested type, the next handler on the stack will be used. The handle to add to the stack.

FileNameEditor class (System.Windows.Forms.Design)

a) *ToString*

### Description

Provides a user interface for editing a filename property.

**System.Windows.Forms.Design.FileNameEditor** provides a file selection dialog for filename selection and editing.

b) *FileNameEditor*

*Example Syntax:*

c) *ToString*

[C#]	public	FileNameEditor();
[C++]	public:	FileNameEditor();
[VB]	Public	Sub New()
[JScript]	public function	FileNameEditor();

d) *EditValue*

[C#]	public	override	object	EditValue(ITypeDescriptorContext	context,	
			IServiceProvider	provider,	object	value);
[C++]	public:	Object*	EditValue(ITypeDescriptorContext*	context,		
		IServiceProvider*	provider,	Object*	value);	

```

1 [VB] Overrides Public Function EditValue(ByVal context As
2 ITypeDescriptorContext, ByVal provider As IServiceProvider, ByVal value As
3 Object) As Object
4 [JScript] public override function EditValue(context : ITypeDescriptorContext,
5 provider : IServiceProvider, value : Object) : Object;
6

```

### *Description*

Edits the specified object using the editor style provided by **System.Windows.Forms.Design.FileNameEditor.GetEditStyle(System.ComponentModel.ITypeDescriptorContext)**.

*Return Value:* The new value of the object. If the value of the object hasn't changed, this should return the same object it was passed.

A service provider is provided so that any required editing services can be obtained. An **System.ComponentModel.ITypeDescriptorContext** that can be used to gain additional context information. A service provider object through which editing services may be obtained. An instance of the value being edited.

### *e) GetEditStyle*

```

14
15
16 [C#] public override UITypeEditorEditStyle
17 GetEditStyle(ITypeDescriptorContext context);
18 [C++] public: UITypeEditorEditStyle GetEditStyle(ITypeDescriptorContext*
19 context);
20 [VB] Overrides Public Function GetEditStyle(ByVal context As
21 ITypeDescriptorContext) As UITypeEditorEditStyle
22 [JScript] public override function GetEditStyle(context : ITypeDescriptorContext)
23 : UITypeEditorEditStyle;
24

```

### *Description*

Gets the editing style of the Edit method.

*Return Value:* A **System.Drawing.Design.UITypeEditorEditStyle** enum value indicating the provided editing style.

If the Edit method is not supported, this will return None. An

**System.ComponentModel.ITypeDescriptorContext** that can be used to gain additional context information.

#### *f) InitializeDialog*

[C#] protected virtual void InitializeDialog(OpenFileDialog openFileDialog);

[C++] protected: virtual void InitializeDialog(OpenFileDialog\* openFileDialog);

[VB] Overridable Protected Sub InitializeDialog(ByVal openFileDialog As OpenFileDialog)

[JScript] protected function InitializeDialog(openFileDialog : OpenFileDialog);

#### *Description*

Initializes the open file dialog when it is created.

This method configures the dialog with a generic file filter and title. An instance of the **System.Windows.Forms.OpenFileDialog** we will use to choose a file name.

FolderNameEditor.FolderBrowser class (System.Windows.Forms.Design)

#### *a) ToString*

#### *Description*

#### *b) FolderNameEditor.FolderBrowser*

*Example Syntax:*

c) *ToString*

```
[C#] public FolderNameEditor.FolderBrowser();  
[C++] public: FolderBrowser();  
[VB] Public Sub New()  
[JScript] public function FolderNameEditor.FolderBrowser();
```

d) *Container*

e) *Description*

f) *ToString*

*Description*

Gets or sets a description to show above the folders. Here you can provide instructions for selecting a folder.

g) *DesignMode*

h) *DirectoryPath*

i) *ToString*

*Description*

Gets the directory path of the folder the user picked.

- j) *Events*
- k) *Site*
- l) *StartLocation*
- m) *ToString*

#### *Description*

Gets/sets the start location of the root node.

- n) *Style*
- o) *ToString*

```
[C#] public FolderNameEditor.FolderBrowserStyles Style {get; set;}
[C++] public: __property FolderNameEditor.FolderBrowserStyles
get_Style();public: __property void
set_Style(FolderNameEditor.FolderBrowserStyles);
[VB] Public Property Style As FolderNameEditor.FolderBrowserStyles
[JScript] public function get Style() :
FolderNameEditor.FolderBrowserStyles;public function set
Style(FolderNameEditor.FolderBrowserStyles);
```

#### *Description*

The styles the folder browser will use when browsing folders. This should be a combination of flags from the FolderBrowserStyles enum.



**p) ShowDialog**

```
[C#]          public          DialogResult          ShowDialog();
[C++]          public:          DialogResult          ShowDialog();
[VB]    Public    Function    ShowDialog()    As    DialogResult
[JavaScript]    public    function    ShowDialog()    :    DialogResult;
```

*Description*

Shows the folder browser dialog.

**q) ShowDialog**

```
[C#]    public    DialogResult    ShowDialog(IWin32Window    owner);
[C++]    public:    DialogResult    ShowDialog(IWin32Window*    owner);
[VB]    Public    Function    ShowDialog(ByVal    owner    As    IWin32Window)    As
DialogResult
[JavaScript]    public    function    ShowDialog(owner : IWin32Window) : DialogResult;
```

*Description*

Shows the folder browser dialog with the specified owner.

**Return Value:** The DialogResult from the form. Top-level window that will own the modal dialog (e.g.: System.Windows.Forms.Form).

FolderNameEditor.FolderBrowserFolder enumeration  
(System.Windows.Forms.Design)

**a) ToString**

*Description*

**b) ToString**

```
[C#]    public    const    FolderNameEditor.FolderBrowserFolder    Desktop;  
[C++]  public:    const    FolderNameEditor.FolderBrowserFolder    Desktop;  
[VB]   Public    Const    Desktop    As    FolderNameEditor.FolderBrowserFolder  
[JScript] public var Desktop : FolderNameEditor.FolderBrowserFolder;
```

*Description*

The user's desktop.

**c) ToString**

```
[C#]    public    const    FolderNameEditor.FolderBrowserFolder    Favorites;  
[C++]  public:    const    FolderNameEditor.FolderBrowserFolder    Favorites;  
[VB]   Public    Const    Favorites    As    FolderNameEditor.FolderBrowserFolder  
[JScript] public var Favorites : FolderNameEditor.FolderBrowserFolder;
```

*Description*

The user's favorites list.

**d) ToString**

```
[C#] public const FolderNameEditor.FolderBrowserFolder MyComputer;  
[C++] public: const FolderNameEditor.FolderBrowserFolder MyComputer;  
[VB] Public Const MyComputer As FolderNameEditor.FolderBrowserFolder  
[JScript] public var MyComputer : FolderNameEditor.FolderBrowserFolder;
```

*Description*

The contents of the My Computer icon.

**e) ToString**

```
[C#] public const FolderNameEditor.FolderBrowserFolder MyDocuments;  
[C++] public: const FolderNameEditor.FolderBrowserFolder MyDocuments;  
[VB] Public Const MyDocuments As FolderNameEditor.FolderBrowserFolder  
[JScript] public var MyDocuments : FolderNameEditor.FolderBrowserFolder;
```

*Description*

The user's My Documents folder.

**f) ToString**

```
[C#] public const FolderNameEditor.FolderBrowserFolder MyPictures;  
[C++] public: const FolderNameEditor.FolderBrowserFolder MyPictures;  
[VB] Public Const MyPictures As FolderNameEditor.FolderBrowserFolder  
[JScript] public var MyPictures : FolderNameEditor.FolderBrowserFolder;
```

*Description*

User's location to store pictures.

*g) ToString*

[C#] public const FolderNameEditor.FolderBrowserFolder

NetAndDialUpConnections;

[C++] public: const FolderNameEditor.FolderBrowserFolder

NetAndDialUpConnections;

[VB] Public Const NetAndDialUpConnections As

FolderNameEditor.FolderBrowserFolder

[JScript] public var NetAndDialUpConnections :

FolderNameEditor.FolderBrowserFolder;

*Description*

Network and dial-up connections.

*h) ToString*

[C#] public const FolderNameEditor.FolderBrowserFolder

NetworkNeighborhood;

[C++] public: const FolderNameEditor.FolderBrowserFolder

NetworkNeighborhood;

[VB] Public Const NetworkNeighborhood As

FolderNameEditor.FolderBrowserFolder

[JScript] public var NetworkNeighborhood :

1 FolderNameEditor.FolderBrowserFolder;

2  
3 *Description*

4 The network neighborhood.

5 *i) ToString*

6  
7 [C#] public const FolderNameEditor.FolderBrowserFolder Printers;

8 [C++] public: const FolderNameEditor.FolderBrowserFolder Printers;

9 [VB] Public Const Printers As FolderNameEditor.FolderBrowserFolder

10 [JScript] public var Printers : FolderNameEditor.FolderBrowserFolder;

11  
12 *Description*

13 A folder containing installed printers.

14 *j) ToString*

15  
16 [C#] public const FolderNameEditor.FolderBrowserFolder Recent;

17 [C++] public: const FolderNameEditor.FolderBrowserFolder Recent;

18 [VB] Public Const Recent As FolderNameEditor.FolderBrowserFolder

19 [JScript] public var Recent : FolderNameEditor.FolderBrowserFolder;

20  
21 *Description*

22 A folder containing shortcuts to recently opened files.

23 *k) ToString*

24  
25 [C#] public const FolderNameEditor.FolderBrowserFolder SendTo;

```

1  [C++]  public:  const  FolderNameEditor.FolderBrowserFolder  SendTo;
2  [VB]   Public  Const  SendTo  As  FolderNameEditor.FolderBrowserFolder
3  [JScript]  public  var  SendTo  :  FolderNameEditor.FolderBrowserFolder;

```

#### *Description*

A folder containing shortcuts to applications to send documents to.

#### *l) ToString*

```

9  [C#]   public  const  FolderNameEditor.FolderBrowserFolder  StartMenu;
10 [C++]  public:  const  FolderNameEditor.FolderBrowserFolder  StartMenu;
11 [VB]   Public  Const  StartMenu  As  FolderNameEditor.FolderBrowserFolder
12 [JScript]  public  var  StartMenu  :  FolderNameEditor.FolderBrowserFolder;

```

#### *Description*

The user's start menu.

#### *m) ToString*

```

18 [C#]   public  const  FolderNameEditor.FolderBrowserFolder  Templates;
19 [C++]  public:  const  FolderNameEditor.FolderBrowserFolder  Templates;
20 [VB]   Public  Const  Templates  As  FolderNameEditor.FolderBrowserFolder
21 [JScript]  public  var  Templates  :  FolderNameEditor.FolderBrowserFolder;

```

#### *Description*

The user's file templates.

FolderNameEditor.FolderBrowserStyles enumeration  
(System.Windows.Forms.Design)

*a) ToString*

*Description*

Specifies various styles that may be applied to the **System.Windows.Forms.Design.FolderNameEditor.FolderBrowser** object.

*b) ToString*

[C#] public const FolderNameEditor.FolderBrowserStyles BrowseForComputer;

[C++] public: const FolderNameEditor.FolderBrowserStyles  
BrowseForComputer;

[VB] Public Const BrowseForComputer As  
FolderNameEditor.FolderBrowserStyles

[JScript] public var BrowseForComputer :  
FolderNameEditor.FolderBrowserStyles;

*Description*

Only return computers. If the user selects anything other than a computer, the **OK** button is grayed.

*c) ToString*

[C#] public const FolderNameEditor.FolderBrowserStyles BrowseForEverything;

[C++] public: const FolderNameEditor.FolderBrowserStyles

1 BrowseForEverything;

2 [VB] Public Const BrowseForEverything As

3 FolderNameEditor.FolderBrowserStyles

4 [JScript] public var BrowseForEverything :

5 FolderNameEditor.FolderBrowserStyles;

6  
7 *Description*

8 Include everything. This will also include files.

9 *d) ToString*

10  
11 [C#] public const FolderNameEditor.FolderBrowserStyles BrowseForPrinter;

12 [C++] public: const FolderNameEditor.FolderBrowserStyles BrowseForPrinter;

13 [VB] Public Const BrowseForPrinter As FolderNameEditor.FolderBrowserStyles

14 [JScript] public var BrowseForPrinter : FolderNameEditor.FolderBrowserStyles;

15  
16 *Description*

17 Only return printers. If the user selects anything other than a printer, the **OK**  
18 button is grayed.

19 *e) ToString*

20  
21 [C#] public const FolderNameEditor.FolderBrowserStyles RestrictToDomain;

22 [C++] public: const FolderNameEditor.FolderBrowserStyles RestrictToDomain;

23 [VB] Public Const RestrictToDomain As FolderNameEditor.FolderBrowserStyles

24 [JScript] public var RestrictToDomain : FolderNameEditor.FolderBrowserStyles;

25



*Description*

Do not include network folders below the domain level in the dialog box's tree view control.

*f) ToString*

[C#] public const FolderNameEditor.FolderBrowserStyles RestrictToFilesystem;

[C++] public: const FolderNameEditor.FolderBrowserStyles  
RestrictToFilesystem;

[VB] Public Const RestrictToFilesystem As  
FolderNameEditor.FolderBrowserStyles

[JScript] public var RestrictToFilesystem :  
FolderNameEditor.FolderBrowserStyles;

*Description*

Only return file system directories. If the user selects folders that are not part of the file system, the **OK** button is grayed.

*g) ToString*

[C#] public const FolderNameEditor.FolderBrowserStyles RestrictToSubfolders;

[C++] public: const FolderNameEditor.FolderBrowserStyles  
RestrictToSubfolders;

[VB] Public Const RestrictToSubfolders As  
FolderNameEditor.FolderBrowserStyles

[JScript] public var RestrictToSubfolders :

FolderNameEditor.FolderBrowserStyles;

### *Description*

Only return file system that is a subfolder of the root folder in the namespace hierarchy. If the user selects a subfolder of the root folder that is not part of the file system, the **OK** button is grayed.

#### *h) ToString*

[C#] public const FolderNameEditor.FolderBrowserStyles ShowTextBox;

[C++] public: const FolderNameEditor.FolderBrowserStyles ShowTextBox;

[VB] Public Const ShowTextBox As FolderNameEditor.FolderBrowserStyles

[JScript] public var ShowTextBox : FolderNameEditor.FolderBrowserStyles;

### *Description*

Includes a textbox control in the browse dialog box that allows the user to type the name of an item.

FolderNameEditor class (System.Windows.Forms.Design)

#### *a) ToString*

### *Description*

Provides a user interface for choosing a folder from the file system.

You may customize the folder options by inheriting from this class and overriding **System.Windows.Forms.Design.FolderNameEditor.InitializeDialog(System.Windows.Forms.Design.FolderNameEditor.FolderBrowser)** .

#### *b) FolderNameEditor*

*Example Syntax:*

c) *ToString*

```
[C#] public FolderNameEditor();
[C++] public: FolderNameEditor();
[VB] Public Sub New()
[JavaScript] public function FolderNameEditor();
```

d) *EditValue*

```
[C#] public override object EditValue(ITypeDescriptorContext context,
IServiceProvider provider, object value);
[C++] public: Object* EditValue(ITypeDescriptorContext* context,
IServiceProvider* provider, Object* value);
[VB] Overrides Public Function EditValue(ByVal context As
ITypeDescriptorContext, ByVal provider As IServiceProvider, ByVal value As
Object) As Object
[JavaScript] public override function EditValue(context : ITypeDescriptorContext,
provider : IServiceProvider, value : Object) : Object;
```

*Description*

Edits the specified object using the editor style provided by **System.Windows.Forms.Design.FolderNameEditor.GetEditStyle(System.ComponentModel.ITypeDescriptorContext)**.

*Return Value:* The new value of the object, or the old value if the object couldn't be updated. An **System.ComponentModel.ITypeDescriptorContext** that can be used to gain additional context information. A service object provider. The value to set.

e) *GetEditStyle*

```
1
2
3 [C#]          public          override          UITypeEditorEditStyle
4 GetEditStyle(ITypeDescriptorContext          context);
5
6 [C++] public: UITypeEditorEditStyle GetEditStyle(ITypeDescriptorContext*
7 context);
8
9 [VB] Overrides Public Function GetEditStyle(ByVal context As
10 ITypeDescriptorContext) As UITypeEditorEditStyle
11
12 [JScript] public override function GetEditStyle(context : ITypeDescriptorContext)
13 :
14 UITypeEditorEditStyle;
```

*Description*

Gets the editing style of the Edit method.

*Return Value:* A **System.Drawing.Design.UITypeEditorEditStyle** enum value indicating the provided editing style.

If the Edit method is not supported, this will return None. An **System.ComponentModel.ITypeDescriptorContext** that can be used to gain additional context information.

f) *InitializeDialog*

```
17
18
19 [C#] protected virtual void InitializeDialog(FolderNameEditor.FolderBrowser
20 folderBrowser);
21
22 [C++] protected: virtual void InitializeDialog(FolderNameEditor.FolderBrowser*
23 folderBrowser);
24
25 [VB] Overridable Protected Sub InitializeDialog(ByVal folderBrowser As
26 FolderNameEditor.FolderBrowser)
27
28 [JScript]          protected          function          InitializeDialog(folderBrowser          :
```

FolderNameEditor.FolderBrowser);

### Description

Initializes the folder browser dialog.

The default implementation provides a generic folder browser. A **System.Windows.Forms.Design.FolderNameEditor.FolderBrowser** to choose a folder.

IMenuEditorService interface (System.Windows.Forms.Design)

#### a) ToString

### Description

Provides access to the menu editing service.

Most of the implementation of this service is provided by the host environment.

#### b) GetMenu

[C#]		Menu		GetMenu();
[C++]		Menu*		GetMenu();
[VB]	Function	GetMenu()	As	Menu
[JScript]	function	GetMenu()	:	Menu;

### Description

Gets the current menu.

#### c) IsActive

[C#]		bool		IsActive();
------	--	------	--	-------------

```

1  [C++]          bool          IsActive();
2  [VB]          Function      IsActive()      As      Boolean
3  [JScript]      function      IsActive()      :      Boolean;
4

```

#### Description

Indicates whether the current menu is active.

#### d) *MessageFilter*

```

9  [C#]          bool          MessageFilter(ref      Message      m);
10 [C++]          bool          MessageFilter(Message*      m);
11 [VB]          Function      MessageFilter(ByRef m As Message) As Boolean
12 [JScript]      function      MessageFilter(m : Message) : Boolean;
13

```

#### Description

Allows the editor service to intercept Win32 messages.

#### e) *SetMenu*

```

18 [C#]          void          SetMenu(Menu      menu);
19 [C++]          void          SetMenu(Menu*      menu);
20 [VB]          Sub          SetMenu(ByVal menu As Menu)
21 [JScript]      function      SetMenu(menu : Menu);
22

```

#### Description

Sets the current menu visible on the form.

The menu will be painted at the top of a form's window frame and can be directly edited by the user. The menu to render.

### *f) SetSelection*

```
[C#]          void          SetSelection(MenuItem          item);
[C++]         void          SetSelection(MenuItem*         item);
[VB]   Sub    SetSelection(ByVal    item    As    MenuItem)
[JScript]     function      SetSelection(item    :    MenuItem);
```

### *Description*

Sets the selected menu item of the current menu.

```
AxImporter.IReferenceResolver          interface
(System.Windows.Forms.Design)
```

### *a) SetSelection*

### *b) ResolveActiveXReference*

```
[C#]   string    ResolveActiveXReference(UCOMITypeLib    typeLib);
[C++]   String*   ResolveActiveXReference(UCOMITypeLib*    typeLib);
[VB]   Function ResolveActiveXReference(ByVal typeLib As UCOMITypeLib) As
String
[JScript] function ResolveActiveXReference(typeLib : UCOMITypeLib) : String;
```

### *c) ResolveComReference*

```
[C#]   string    ResolveComReference(AssemblyName    name);
[C++]   String*   ResolveComReference(AssemblyName*    name);
```

1 [VB] Function ResolveComReference(ByVal name As AssemblyName) As String

2 [JScript] function ResolveComReference(name : AssemblyName) : String;

3 *d) ResolveComReference*

5 [C#] string ResolveComReference(UCOMITypeLib typeLib);

6 [C++] String\* ResolveComReference(UCOMITypeLib\* typeLib);

7 [VB] Function ResolveComReference(ByVal typeLib As UCOMITypeLib) As  
8 String

9 [JScript] function ResolveComReference(typeLib : UCOMITypeLib) : String;

10 *e) ResolveManagedReference*

12 [C#] string ResolveManagedReference(string assemName);

13 [C++] String\* ResolveManagedReference(String\* assemName);

14 [VB] Function ResolveManagedReference(ByVal assemName As String) As  
15 String

16 [JScript] function ResolveManagedReference(assemName : String) : String;

17 MenuCommands class (System.Windows.Forms.Design)

18 *a) ResolveManagedReference*

21 *Description*

22 Provides command IDs and GUIDS that correspond to the host Command Bar  
23 menu layout.



**b) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    ComponentTrayMenu;  
[C++]    public:    static    CommandID*    ComponentTrayMenu;  
[VB]    Public    Shared    ReadOnly    ComponentTrayMenu    As    CommandID  
[JScript]    public    static    var    ComponentTrayMenu    :    CommandID;
```

*Description*

**c) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    ContainerMenu;  
[C++]    public:    static    CommandID*    ContainerMenu;  
[VB]    Public    Shared    ReadOnly    ContainerMenu    As    CommandID  
[JScript]    public    static    var    ContainerMenu    :    CommandID;
```

*Description*

**d) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    DesignerProperties;  
[C++]    public:    static    CommandID*    DesignerProperties;  
[VB]    Public    Shared    ReadOnly    DesignerProperties    As    CommandID  
[JScript]    public    static    var    DesignerProperties    :    CommandID;
```

*Description*

*e) ResolveManagedReference*

[C#]	public	static	readonly	CommandID	KeyCancel;
[C++]	public:	static		CommandID*	KeyCancel;
[VB]	Public	Shared	ReadOnly	KeyCancel	As CommandID
[JScript]	public	static	var	KeyCancel	: CommandID;

*Description*

*f) ResolveManagedReference*

[C#]	public	static	readonly	CommandID	KeyDefaultAction;
[C++]	public:	static		CommandID*	KeyDefaultAction;
[VB]	Public	Shared	ReadOnly	KeyDefaultAction	As CommandID
[JScript]	public	static	var	KeyDefaultAction	: CommandID;

*Description*

*g) ResolveManagedReference*

[C#]	public	static	readonly	CommandID	KeyMoveDown;
[C++]	public:	static		CommandID*	KeyMoveDown;

1 [VB] Public Shared ReadOnly KeyMoveDown As CommandID

2 [JScript] public static var KeyMoveDown : CommandID;

4 *Description*

6 ***h) ResolveManagedReference***

8 [C#] public static readonly CommandID KeyMoveLeft;

9 [C++] public: static CommandID\* KeyMoveLeft;

10 [VB] Public Shared ReadOnly KeyMoveLeft As CommandID

11 [JScript] public static var KeyMoveLeft : CommandID;

13 *Description*

15 ***i) ResolveManagedReference***

17 [C#] public static readonly CommandID KeyMoveRight;

18 [C++] public: static CommandID\* KeyMoveRight;

19 [VB] Public Shared ReadOnly KeyMoveRight As CommandID

20 [JScript] public static var KeyMoveRight : CommandID;

22 *Description*

**j) *ResolveManagedReference***

```
[C#]      public      static      readonly      CommandID      KeyMoveUp;
[C++]      public:      static      CommandID*      KeyMoveUp;
[VB]      Public      Shared      ReadOnly      KeyMoveUp      As      CommandID
[JScript]      public      static      var      KeyMoveUp      :      CommandID;
```

*Description*

**k) *ResolveManagedReference***

```
[C#]      public      static      readonly      CommandID      KeyNudgeDown;
[C++]      public:      static      CommandID*      KeyNudgeDown;
[VB]      Public      Shared      ReadOnly      KeyNudgeDown      As      CommandID
[JScript]      public      static      var      KeyNudgeDown      :      CommandID;
```

*Description*

**l) *ResolveManagedReference***

```
[C#]      public      static      readonly      CommandID      KeyNudgeHeightDecrease;
[C++]      public:      static      CommandID*      KeyNudgeHeightDecrease;
[VB]      Public      Shared      ReadOnly      KeyNudgeHeightDecrease      As      CommandID
[JScript]      public      static      var      KeyNudgeHeightDecrease      :      CommandID;
```

*Description*

**m) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeyNudgeHeightIncrease;
[C++]   public:    static    CommandID*    KeyNudgeHeightIncrease;
[VB]    Public    Shared    ReadOnly    KeyNudgeHeightIncrease    As    CommandID
[JScript] public    static    var    KeyNudgeHeightIncrease    :    CommandID;
```

*Description*

**n) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeyNudgeLeft;
[C++]   public:    static    CommandID*    KeyNudgeLeft;
[VB]    Public    Shared    ReadOnly    KeyNudgeLeft    As    CommandID
[JScript] public    static    var    KeyNudgeLeft    :    CommandID;
```

*Description*

**o) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeyNudgeRight;
[C++]   public:    static    CommandID*    KeyNudgeRight;
```

1 [VB] Public Shared ReadOnly KeyNudgeRight As CommandID

2 [JScript] public static var KeyNudgeRight : CommandID;

4 *Description*

6 **p) *ResolveManagedReference***

8 [C#] public static readonly CommandID KeyNudgeUp;

9 [C++] public: static CommandID\* KeyNudgeUp;

10 [VB] Public Shared ReadOnly KeyNudgeUp As CommandID

11 [JScript] public static var KeyNudgeUp : CommandID;

13 *Description*

15 **q) *ResolveManagedReference***

17 [C#] public static readonly CommandID KeyNudgeWidthDecrease;

18 [C++] public: static CommandID\* KeyNudgeWidthDecrease;

19 [VB] Public Shared ReadOnly KeyNudgeWidthDecrease As CommandID

20 [JScript] public static var KeyNudgeWidthDecrease : CommandID;

22 *Description*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

**r) *ResolveManagedReference***

```
[C#] public static readonly CommandID KeyNudgeWidthIncrease;  
[C++] public: static CommandID* KeyNudgeWidthIncrease;  
[VB] Public Shared ReadOnly KeyNudgeWidthIncrease As CommandID  
[JScript] public static var KeyNudgeWidthIncrease : CommandID;
```

*Description*

**s) *ResolveManagedReference***

```
[C#] public static readonly CommandID KeyReverseCancel;  
[C++] public: static CommandID* KeyReverseCancel;  
[VB] Public Shared ReadOnly KeyReverseCancel As CommandID  
[JScript] public static var KeyReverseCancel : CommandID;
```

*Description*

**t) *ResolveManagedReference***

```
[C#] public static readonly CommandID KeySelectNext;  
[C++] public: static CommandID* KeySelectNext;  
[VB] Public Shared ReadOnly KeySelectNext As CommandID  
[JScript] public static var KeySelectNext : CommandID;
```

*Description*

**u) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeySelectPrevious;
[C++]    public:    static    CommandID*    KeySelectPrevious;
[VB]    Public    Shared    ReadOnly    KeySelectPrevious    As    CommandID
[JScript]    public    static    var    KeySelectPrevious    :    CommandID;
```

*Description*

**v) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeySizeHeightDecrease;
[C++]    public:    static    CommandID*    KeySizeHeightDecrease;
[VB]    Public    Shared    ReadOnly    KeySizeHeightDecrease    As    CommandID
[JScript]    public    static    var    KeySizeHeightDecrease    :    CommandID;
```

*Description*

**w) *ResolveManagedReference***

```
[C#]    public    static    readonly    CommandID    KeySizeHeightIncrease;
[C++]    public:    static    CommandID*    KeySizeHeightIncrease;
```



1 [VB] Public Shared ReadOnly KeySizeHeightIncrease As CommandID

2 [JScript] public static var KeySizeHeightIncrease : CommandID;

4 *Description*

6 *x) ResolveManagedReference*

8 [C#] public static readonly CommandID KeySizeWidthDecrease;

9 [C++] public: static CommandID\* KeySizeWidthDecrease;

10 [VB] Public Shared ReadOnly KeySizeWidthDecrease As CommandID

11 [JScript] public static var KeySizeWidthDecrease : CommandID;

13 *Description*

15 *y) ResolveManagedReference*

17 [C#] public static readonly CommandID KeySizeWidthIncrease;

18 [C++] public: static CommandID\* KeySizeWidthIncrease;

19 [VB] Public Shared ReadOnly KeySizeWidthIncrease As CommandID

20 [JScript] public static var KeySizeWidthIncrease : CommandID;

22 *Description*

z) *ResolveManagedReference*

```
[C#]    public    static    readonly    CommandID    KeyTabOrderSelect;
[C++]    public:    static    CommandID*    KeyTabOrderSelect;
[VB]    Public    Shared    ReadOnly    KeyTabOrderSelect    As    CommandID
[JScript]    public    static    var    KeyTabOrderSelect    :    CommandID;
```

*Description*

aa) *ResolveManagedReference*

```
[C#]    public    static    readonly    CommandID    SelectionMenu;
[C++]    public:    static    CommandID*    SelectionMenu;
[VB]    Public    Shared    ReadOnly    SelectionMenu    As    CommandID
[JScript]    public    static    var    SelectionMenu    :    CommandID;
```

*Description*

bb) *ResolveManagedReference*

```
[C#]    public    static    readonly    CommandID    TraySelectionMenu;
[C++]    public:    static    CommandID*    TraySelectionMenu;
[VB]    Public    Shared    ReadOnly    TraySelectionMenu    As    CommandID
[JScript]    public    static    var    TraySelectionMenu    :    CommandID;
```

*Description*

*cc) MenuCommands*

*Example Syntax:*

*dd) ResolveManagedReference*

[C#]	public	MenuCommands();
[C++]	public:	MenuCommands();
[VB]	Public	Sub New()
[JScript]	public function	MenuCommands();

AxImporter.Options class (System.Windows.Forms.Design)

a) *ToString*

b) *ToString*

c) *ToString*

d) *ToString*

e) *ToString*

f) *ToString*

g) *ToString*

h) *ToString*

i) *ToString*

j) *ToString*

k) *ToString*

l) *ToString*

m) *ToString*

n) *ToString*

o) *AxImporter.Options*

*Example Syntax:*

p) *ToString*

ParentControlDesigner class (System.Windows.Forms.Design)

a) *ToString*

*Description*

Provides the design-time functionality of the **System.Windows.Forms.Design.ControlDesigner** , as well as a selection UI handler and the ability to manipulate child components.

b) *ParentControlDesigner*

*Example Syntax:*

c) *ToString*

d) *AccessibilityObject*

e) *AssociatedComponents*

f) *Component*

g) *Control*

h) *DefaultControlLocation*

i) *ToString*

*Description*

Indicates the default location for a control added to this designer.

The default location is usually (0,0), but may be modified if the container has special borders, etc.

j) *DrawGrid*

k) *ToString*

[C#]      protected      virtual      bool      DrawGrid      {get;      set;}

[C++] protected: \_\_property virtual bool get\_DrawGrid();protected: \_\_property

virtual                                  void                                  set\_DrawGrid(bool);

[VB]      Overridable      Protected      Property      DrawGrid      As      Boolean

[JScript] protected function get DrawGrid() : Boolean;protected function set DrawGrid(Boolean);

#### *Description*

Accessor method for the DrawGrid property. This property determines if the grid should be drawn on a control.

*l) EnableDragRect*

*m) ToString*

[C#] protected override bool EnableDragRect {get;}

[C++] protected: \_\_property virtual bool get\_EnableDragRect();

[VB] Overrides Protected ReadOnly Property EnableDragRect As Boolean

[JScript] protected function get EnableDragRect() : Boolean;

#### *Description*

Indicates whether drag rectangles can be drawn on this designer.

*n) GridSize*

*o) ToString*

[C#] protected Size GridSize {get; set;}

[C++] protected: \_\_property Size get\_GridSize();protected: \_\_property void set\_GridSize(Size);

[VB] Protected Property GridSize As Size

[JScript] protected function get GridSize() : Size;protected function set GridSize(Size);

## Description

Gets/Sets the GridSize property for a form or user control.

*Return Value:* A Point representing the size of the grid drawn on a form or user control.

*p) InheritanceAttribute*

*q) Inherited*

*r) SelectionRules*

*s) ShadowProperties*

*t) Verbs*

*u) CanParent*

[C#] public virtual bool CanParent(Control control);

[C++] public: virtual bool CanParent(Control\* control);

[VB] Overridable Public Function CanParent(ByVal control As Control) As Boolean

[JScript] public function CanParent(control : Control) : Boolean; Indicates whether this designer can parent the specified control.

## Description

Indicates whether this designer can parent the specified control.

*Return Value:* **true** if this designer can parent the specified control; otherwise, **false**.

*v) CanParent*

[C#] public virtual bool CanParent(ControlDesigner controlDesigner);

```

1 [C++] public: virtual bool CanParent(ControlDesigner* controlDesigner);
2 [VB] Overridable Public Function CanParent(ByVal controlDesigner As
3 ControlDesigner) As Boolean
4 [JScript] public function CanParent(controlDesigner : ControlDesigner) : Boolean;
5 Indicates whether this designer can parent to the specified designer.
6

```

### *Description*

Indicates whether this designer can parent to the specified designer.

Generally this means if the control for this designer can parent the specified ControlDesigner's designer. The designer to determine whether this designer can parent.

### *w) CreateTool*

```

13 [C#] protected void CreateTool(ToolboxItem tool);
14 [C++] protected: void CreateTool(ToolboxItem* tool);
15 [VB] Protected Sub CreateTool(ByVal tool As ToolboxItem)
16 [JScript] protected function CreateTool(tool : ToolboxItem); Creates the specified
17 tool in the center of the currently selected control.
18

```

### *Description*

Creates the specified tool in the center of the currently selected control.

The default size for the tool is used. The tool to create.

### *x) CreateTool*

```

24 [C#] protected void CreateTool(ToolboxItem tool, Point location);
25 [C++] protected: void CreateTool(ToolboxItem* tool, Point location);

```



[VB] Protected Sub CreateTool(ByVal tool As ToolboxItem, ByVal location As Point)

[JScript] protected function CreateTool(tool : ToolboxItem, location : Point);

Creates the specified tool in the currently selected control at the specified position.

#### *Description*

Creates the specified tool in the currently selected control at the specified position.

The default size for the tool is used. The tool to create. The location, in screen coordinates, of the tool's upper left corner. If the tool has a default size it will be centered at this location.

#### *y) CreateTool*

[C#] protected void CreateTool(ToolboxItem tool, Rectangle bounds);

[C++] protected: void CreateTool(ToolboxItem\* tool, Rectangle bounds);

[VB] Protected Sub CreateTool(ByVal tool As ToolboxItem, ByVal bounds As Rectangle)

[JScript] protected function CreateTool(tool : ToolboxItem, bounds : Rectangle);

#### *Description*

Creates the specified tool in the currently selected control.

The tool is created with the provided shape. The tool to create. The dimensions of the tool. The x and y components of the bounds should be the screen coordinates of the upper left corner of the tool.

#### *z) CreateToolCore*

[C#] protected virtual IComponent[] CreateToolCore(ToolboxItem tool, int x, int

```

1 y, int width, int height, bool hasLocation, bool hasSize);
2 [C++] protected: virtual IComponent* CreateToolCore(ToolboxItem* tool, int x,
3 int y, int width, int height, bool hasLocation, bool hasSize) [];
4 [VB] Overridable Protected Function CreateToolCore(ByVal tool As
5 ToolboxItem, ByVal x As Integer, ByVal y As Integer, ByVal width As Integer,
6 ByVal height As Integer, ByVal hasLocation As Boolean, ByVal hasSize As
7 Boolean) As IComponent()
8 [JScript] protected function CreateToolCore(tool : ToolboxItem, x : int, y : int,
9 width : int, height : int, hasLocation : Boolean, hasSize : Boolean) : IComponent[];

```

### *Description*

Provides core functionality for all the CreateTool methods.

*Return Value:* An array of components that were created by the tool.

This is the only CreateTool method that can be overridden. The tool to create. The x location, in screen coordinates, of the location of the tool. If the size of the tool is specified here, this will be the upper left corner of the tool. If no size is specified, then this should be interpreted as the center of the tool. The y location, in screen coordinates, of the location of the tool. If the size of the tool is specified here, this will be the upper left corner of the tool. If no size is specified, then this should be interpreted as the center of the tool. The width of the tool. This is ignored if hasSize is false. The height of the tool. This is ignored if hasSize is false. If this is false the x and y values will be calculated to place the tool in the center of the currently selected control. If this is false the width and height values will be calculated to be the default width and height for the control.

### *aa) Dispose*

```

22 [C#] protected override void Dispose(bool disposing);
23 [C++] protected: void Dispose(bool disposing);
24 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)
25

```

[JScript] protected override function Dispose(disposing : Boolean);

### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.Design.ParentControlDesigner**.

### *Call*

**System.Windows.Forms.Design.ParentControlDesigner.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.Design.ParentControlDesigner**. The **System.Windows.Forms.Design.ParentControlDesigner.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.Design.ParentControlDesigner** in an unusable state. After calling **System.Windows.Forms.Design.ParentControlDesigner.Dispose(System.Boolean)**, you must release all references to the **System.Windows.Forms.Design.ParentControlDesigner** so the memory it was occupying can be reclaimed by garbage collection.

### *bb) GetControl*

[C#] protected Control GetControl(object component);

[C++] protected: Control\* GetControl(Object\* component);

[VB] Protected Function GetControl(ByVal component As Object) As Control

[JScript] protected function GetControl(component : Object) : Control; Returns the control that represents the UI for the given component.

### *cc) GetUpdatedRect*

[C#] protected Rectangle GetUpdatedRect(Rectangle originalRect, Rectangle dragRect, bool updateSize);

[C++] protected: Rectangle GetUpdatedRect(Rectangle originalRect, Rectangle dragRect, bool updateSize);

```

1 [VB] Protected Function GetUpdatedRect(ByVal originalRect As Rectangle,
2   ByVal dragRect As Rectangle, ByVal updateSize As Boolean) As Rectangle
3 [JScript] protected function GetUpdatedRect(originalRect : Rectangle, dragRect :
4   Rectangle,          updateSize          : Boolean)          : Rectangle;

```

### *Description*

Updates the given rectangle, adjusting it for grid snaps as needed.

*Return Value:* the newly updated rectangle. If no changes were needed to the rectangle this will return rect. The original rectangle of the component being updated. The dragging rectangle of the component. True to updateSize as well

### *dd) Initialize*

```

11 [C#]    public    override    void    Initialize(IComponent    component);
12 [C++]    public:    void    Initialize(IComponent*    component);
13 [VB] Overrides Public Sub Initialize(ByVal component As IComponent)
14 [JScript] public override function Initialize(component : IComponent);

```

### *Description*

Initializes the designer with the given component. The designer can get the component's site and request services from it in this call. The component to associate with this designer.

### *ee) InvokeCreateTool*

```

21
22 [C#] protected static void InvokeCreateTool(ParentControlDesigner toInvoke,
23   ToolboxItem                                                    tool);
24 [C++] protected: static void InvokeCreateTool(ParentControlDesigner* toInvoke,
25   ToolboxItem*                                                    tool);

```

```

1 [VB] Protected Shared Sub InvokeCreateTool(ByVal toInvoke As
2 ParentControlDesigner, ByVal tool As ToolboxItem)
3 [JScript] protected static function InvokeCreateTool(toInvoke :
4 ParentControlDesigner, tool : ToolboxItem);

```

## *Description*

### *ff) OnDragDrop*

```

10 [C#] protected override void OnDragDrop(DragEventArgs de);
11 [C++] protected: void OnDragDrop(DragEventArgs* de);
12 [VB] Overrides Protected Sub OnDragDrop(ByVal de As DragEventArgs)
13 [JScript] protected override function OnDragDrop(de : DragEventArgs);

```

## *Description*

Called in response to a drag drop for OLE drag and drop. Here we drop a toolbox component on our parent control.

### *gg) OnDragEnter*

```

19 [C#] protected override void OnDragEnter(DragEventArgs de);
20 [C++] protected: void OnDragEnter(DragEventArgs* de);
21 [VB] Overrides Protected Sub OnDragEnter(ByVal de As DragEventArgs)
22 [JScript] protected override function OnDragEnter(de : DragEventArgs);

```

## *Description*

Called in response to a drag enter for OLE drag and drop.

**hh) OnDragLeave**

```
[C#]    protected    override    void    OnDragLeave(EventArgs    e);
[C++]    protected:    void    OnDragLeave(EventArgs*    e);
[VB]    Overrides    Protected    Sub    OnDragLeave(ByVal e As EventArgs)
[JScript]    protected    override    function    OnDragLeave(e : EventArgs);
```

*Description*

Called when a drag-drop operation leaves the control designer view  
Called when a drag-drop operation leaves the control designer view

**ii) OnDragOver**

```
[C#]    protected    override    void    OnDragOver(DragEventArgs    de);
[C++]    protected:    void    OnDragOver(DragEventArgs*    de);
[VB]    Overrides    Protected    Sub    OnDragOver(ByVal de As DragEventArgs)
[JScript]    protected    override    function    OnDragOver(de : DragEventArgs);
```

*Description*

Called when a drag drop object is dragged over the control designer view  
Called when a drag drop object is dragged over the control designer view

**jj) OnGiveFeedback**

```
[C#]    protected    override    void    OnGiveFeedback(GiveFeedbackEventArgs    e);
[C++]    protected:    void    OnGiveFeedback(GiveFeedbackEventArgs*    e);
[VB]    Overrides    Protected    Sub    OnGiveFeedback(ByVal e As
```

GiveFeedbackEventArgs)

```
[JScript]    protected    override    function    OnGiveFeedback(e :  
GiveFeedbackEventArgs);
```

### *Description*

Event handler for our GiveFeedback event, which is called when a drag operation is in progress. The host will call us with this when an OLE drag event happens.

### *kk) OnMouseDownBegin*

```
[C#]    protected    override    void    OnMouseDownBegin(int    x,    int    y);  
[C++]    protected:    void    OnMouseDownBegin(int    x,    int    y);  
[VB] Overrides Protected Sub OnMouseDownBegin(ByVal x As Integer, ByVal y  
As Integer)  
[JScript] protected override function OnMouseDownBegin(x : int, y : int);
```

### *Description*

Called in response to the left mouse button being pressed on a component. The designer overrides this to provide a "lasso" selection for components within the control. The x position, in control coordinates, of the mouse. The y position, in control coordinates, of the mouse.

### *ll) OnMouseDownEnd*

```
[C#]    protected    override    void    OnMouseDownEnd(bool    cancel);  
[C++]    protected:    void    OnMouseDownEnd(bool    cancel);  
[VB] Overrides Protected Sub OnMouseDownEnd(ByVal cancel As Boolean)  
[JScript] protected override function OnMouseDownEnd(cancel : Boolean);
```

## Description

Called at the end of a drag operation. This either commits or rolls back the drag. Set this to true to cancel the drag, or false to commit it.

### *mm) OnMouseDownMove*

[C#] protected override void OnMouseDownMove(int x, int y);

[C++] protected: void OnMouseDownMove(int x, int y);

[VB] Overrides Protected Sub OnMouseDownMove(ByVal x As Integer, ByVal y As Integer)

[JScript] protected override function OnMouseDownMove(x : int, y : int);

## Description

Called for each movement of the mouse. This will check to see if a drag operation is in progress. If so, it will pass the updated drag dimensions on to the selection UI service. The x position, in screen coordinates, of the mouse. The y position, in screen coordinates, of the mouse.

### *nn) OnMouseEnter*

[C#] protected override void OnMouseEnter();

[C++] protected: void OnMouseEnter();

[VB] Overrides Protected Sub OnMouseEnter()

[JScript] protected override function OnMouseEnter();

## Description

Called when the mouse enters the control. At this point we want to cancel the timeout timer.



**oo) OnMouseHover**

[C#]	protected	override	void	OnMouseHover();
[C++]	protected:		void	OnMouseHover();
[VB]	Overrides	Protected	Sub	OnMouseHover()
[JScript]	protected	override	function	OnMouseHover();

*Description*

Called when the user hovers over the control. At this point we want to display the container selector. Note: Since the child controls pass this notification up, we will display the container selector even if you hover over a child control.

**pp) OnMouseLeave**

[C#]	protected	override	void	OnMouseLeave();
[C++]	protected:		void	OnMouseLeave();
[VB]	Overrides	Protected	Sub	OnMouseLeave()
[JScript]	protected	override	function	OnMouseLeave();

*Description*

Called when the mouse leaves the control. At this point we need to start listening for the container selector timeout.

**qq) OnPaintAdornments**

[C#]	protected	override	void	OnPaintAdornments(PaintEventArgs pe);
[C++]	protected:		void	OnPaintAdornments(PaintEventArgs* pe);
[VB]	Overrides	Protected	Sub	OnPaintAdornments(ByVal pe As PaintEventArgs)

[JScript] protected override function OnPaintAdornments(pe : PaintEventArgs);

### *Description*

Called after our component has finished painting. Here we draw our grid surface  
Called after our component has finished painting. Here we draw our grid surface  
The paint event.

### *rr) OnSetCursor*

[C#] protected override void OnSetCursor();

[C++] protected: void OnSetCursor();

[VB] Overrides Protected Sub OnSetCursor()

[JScript] protected override function OnSetCursor();

### *Description*

Called each time the cursor needs to be set. The ParentControlDesigner behavior here will set the cursor to one of three things: 1. If the toolbox service has a tool selected, it will allow the toolbox service to set the cursor. 2. The arrow will be set. Parent controls allow dragging within their interior.

### *ss) PreFilterProperties*

[C#] protected override void PreFilterProperties(IDictionary properties);

[C++] protected: void PreFilterProperties(IDictionary\* properties);

[VB] Overrides Protected Sub PreFilterProperties(ByVal properties As  
IDictionary)

[JScript] protected override function PreFilterProperties(properties : IDictionary);

### *Description*

Allows a designer to filter the set of properties the component it is designing will expose through the TypeDescriptor object. This method is called immediately before its corresponding "Post" method. If you are overriding this method you should call the base implementation before you perform your own filtering.  
*Return Value:* The augmented set of properties. If the method does not modify any properties, it may just return a reference to its input parameter. If you do make a change to the properties, you must create a new array. The properties for the class of the component.

**tt) IOleDragClient.AddComponent**

[C#] bool IOleDragClient.AddComponent(IComponent component, string name, bool firstAdd);

[C++] bool IOleDragClient::AddComponent(IComponent\* component, String\* name, bool firstAdd);

[VB] Function AddComponent(ByVal component As IComponent, ByVal name As String, ByVal firstAdd As Boolean) As Boolean Implements IOleDragClient.AddComponent

[JScript] function IOleDragClient.AddComponent(component : IComponent, name : String, firstAdd : Boolean) : Boolean;

**uu) IOleDragClient.GetControlForComponent**

[C#] Control IOleDragClient.GetControlForComponent(object component);

[C++] Control\* IOleDragClient::GetControlForComponent(Object\* component);

[VB] Function GetControlForComponent(ByVal component As Object) As Control Implements IOleDragClient.GetControlForComponent

[JScript] function IOleDragClient.GetControlForComponent(component : Object) : Control;

vv) *IOleDragClient.GetDesignerControl*

```
1
2
3 [C#]          Control          IOleDragClient.GetDesignerControl();
4
5 [C++]          Control*          IOleDragClient::GetDesignerControl();
6
7 [VB]  Function  GetDesignerControl()  As  Control  Implements
8      IOleDragClient.GetDesignerControl
9
10 [JScript] function IOleDragClient.GetDesignerControl() : Control;
```

ww) *IOleDragClient.IsDropOk*

```
9
10 [C#]    bool    IOleDragClient.IsDropOk(IComponent    component);
11
12 [C++]    bool    IOleDragClient::IsDropOk(IComponent*    component);
13
14 [VB]  Function  IsDropOk(ByVal component As IComponent) As Boolean
15      Implements                                  IOleDragClient.IsDropOk
16
17 [JScript] function IOleDragClient.IsDropOk(component : IComponent) : Boolean;
```

xx) *ISelectionUIHandler.BeginDrag*

```
16
17 [C#] bool ISelectionUIHandler.BeginDrag(object[] components, SelectionRules
18 rules,          int          initialX,          int          initialY);
19
20 [C++] bool ISelectionUIHandler::BeginDrag(Object* components __gc[],
21 SelectionRules rules,          int          initialX,          int          initialY);
22
23 [VB] Function BeginDrag(ByVal components() As Object, ByVal rules As
24 SelectionRules, ByVal initialX As Integer, ByVal initialY As Integer) As Boolean
25      Implements                                  ISelectionUIHandler.BeginDrag
26
27 [JScript] function ISelectionUIHandler.BeginDrag(components : Object[], rules :
28 SelectionRules, initialX : int, initialY : int) : Boolean;
```

yy) *ISelectionUIHandler.DragMoved*

[C#] void ISelectionUIHandler.DragMoved(object[] components, Rectangle offset);

[C++] void ISelectionUIHandler::DragMoved(Object\* components \_\_gc[], Rectangle offset);

[VB] Sub DragMoved(ByVal components() As Object, ByVal offset As Rectangle) Implements ISelectionUIHandler.DragMoved

[JScript] function ISelectionUIHandler.DragMoved(components : Object[], offset : Rectangle);

zz) *ISelectionUIHandler.EndDrag*

[C#] void ISelectionUIHandler.EndDrag(object[] components, bool cancel);

[C++] void ISelectionUIHandler::EndDrag(Object\* components \_\_gc[], bool cancel);

[VB] Sub EndDrag(ByVal components() As Object, ByVal cancel As Boolean) Implements ISelectionUIHandler.EndDrag

[JScript] function ISelectionUIHandler.EndDrag(components : Object[], cancel : Boolean);

aaa) *ISelectionUIHandler.GetComponentBounds*

[C#] Rectangle ISelectionUIHandler.GetComponentBounds(object component);

[C++] Rectangle ISelectionUIHandler::GetComponentBounds(Object\* component);

[VB] Function GetComponentBounds(ByVal component As Object) As Rectangle

1 Implements ISelectionUIHandler.GetComponentBounds

2 [JScript] function ISelectionUIHandler.GetComponentBounds(component :  
3 Object) : Rectangle;

4 ***bbb) ISelectionUIHandler.GetComponentRules***

6 [C#] SelectionRules ISelectionUIHandler.GetComponentRules(object  
7 component);

8 [C++] SelectionRules ISelectionUIHandler::GetComponentRules(Object\*  
9 component);

10 [VB] Function GetComponentRules(ByVal component As Object) As  
11 SelectionRules Implements ISelectionUIHandler.GetComponentRules  
12 [JScript] function ISelectionUIHandler.GetComponentRules(component : Object)  
13 : SelectionRules;

14 ***ccc) ISelectionUIHandler.GetSelectionClipRect***

16 [C#] Rectangle ISelectionUIHandler.GetSelectionClipRect(object component);

17 [C++] Rectangle ISelectionUIHandler::GetSelectionClipRect(Object\*  
18 component);

19 [VB] Function GetSelectionClipRect(ByVal component As Object) As Rectangle  
20 Implements ISelectionUIHandler.GetSelectionClipRect

21 [JScript] function ISelectionUIHandler.GetSelectionClipRect(component : Object)  
22 : Rectangle;

**ddd) ISelectionUIHandler.OleDragDrop**

```
[C#]    void    ISelectionUIHandler.OleDragDrop(DragEventArgs    de);
[C++]    void    ISelectionUIHandler::OleDragDrop(DragEventArgs*    de);
[VB]    Sub    OleDragDrop(ByVal de As DragEventArgs) Implements
ISelectionUIHandler.OleDragDrop
[JScript] function ISelectionUIHandler.OleDragDrop(de : DragEventArgs);
```

**eee) ISelectionUIHandler.OleDragEnter**

```
[C#]    void    ISelectionUIHandler.OleDragEnter(DragEventArgs    de);
[C++]    void    ISelectionUIHandler::OleDragEnter(DragEventArgs*    de);
[VB]    Sub    OleDragEnter(ByVal de As DragEventArgs) Implements
ISelectionUIHandler.OleDragEnter
[JScript] function ISelectionUIHandler.OleDragEnter(de : DragEventArgs);
```

**fff) ISelectionUIHandler.OleDragLeave**

```
[C#]    void    ISelectionUIHandler.OleDragLeave();
[C++]    void    ISelectionUIHandler::OleDragLeave();
[VB]    Sub    OleDragLeave() Implements ISelectionUIHandler.OleDragLeave
[JScript] function ISelectionUIHandler.OleDragLeave();
```

**ggg) ISelectionUIHandler.OleDragOver**

```
[C#]    void    ISelectionUIHandler.OleDragOver(DragEventArgs    de);
[C++]    void    ISelectionUIHandler::OleDragOver(DragEventArgs*    de);
[VB]    Sub    OleDragOver(ByVal de As DragEventArgs) Implements
```

1 ISelectionUIHandler.OleDragOver

2 [JScript] function ISelectionUIHandler.OleDragOver(de : DragEventArgs);

3 ***hhh) ISelectionUIHandler.OnSelectionDoubleClick***

4  
5 [C#] void ISelectionUIHandler.OnSelectionDoubleClick(IComponent  
6 component);

7 [C++] void ISelectionUIHandler::OnSelectionDoubleClick(IComponent\*  
8 component);

9 [VB] Sub OnSelectionDoubleClick(ByVal component As IComponent)

10 Implements ISelectionUIHandler.OnSelectionDoubleClick

11 [JScript] function ISelectionUIHandler.OnSelectionDoubleClick(component :  
12 IComponent);

13 ***iii) ISelectionUIHandler.QueryBeginDrag***

14  
15 [C#] bool ISelectionUIHandler.QueryBeginDrag(object[] components,  
16 SelectionRules rules, int initialX, int initialY);

17 [C++] bool ISelectionUIHandler::QueryBeginDrag(Object\* components \_\_gc[],  
18 SelectionRules rules, int initialX, int initialY);

19 [VB] Function QueryBeginDrag(ByVal components() As Object, ByVal rules As  
20 SelectionRules, ByVal initialX As Integer, ByVal initialY As Integer) As Boolean

21 Implements ISelectionUIHandler.QueryBeginDrag

22 [JScript] function ISelectionUIHandler.QueryBeginDrag(components : Object[],  
23 rules : SelectionRules, initialX : int, initialY : int) : Boolean;



**jjj) ISelectionUIHandler.ShowContextMenu**

[C#] void ISelectionUIHandler.ShowContextMenu(IComponent component);  
[C++] void ISelectionUIHandler::ShowContextMenu(IComponent\* component);  
[VB] Sub ShowContextMenu(ByVal component As IComponent) Implements  
ISelectionUIHandler.ShowContextMenu  
[JScript] function ISelectionUIHandler.ShowContextMenu(component :  
IComponent);

**kkk) WndProc**

[C#] protected override void WndProc(ref Message m);  
[C++] protected: void WndProc(Message\* m);  
[VB] Overrides Protected Sub WndProc(ByRef m As Message)  
[JScript] protected override function WndProc(m : Message);

*Description*

Provides processing for Windows messages.

ScrollableControlDesigner class (System.Windows.Forms.Design)

**a) WndProc**

*Description*

Provides design-time functionality for scrollable controls.

**b) ScrollableControlDesigner**

*Example Syntax:*

- c) *WndProc*
- d) *AccessibilityObject*
- e) *AssociatedComponents*
- f) *Component*
- g) *Control*
- h) *DefaultControlLocation*
- i) *DrawGrid*
- j) *EnableDragRect*
- k) *GridSize*
- l) *InheritanceAttribute*
- m) *Inherited*
- n) *SelectionRules*
- o) *ShadowProperties*
- p) *Verbs*
- q) *GetHitTest*

[C#]        protected        override        bool        GetHitTest(Point        pt);

[C++]        protected:        bool        GetHitTest(Point        pt);

[VB] Overrides Protected Function GetHitTest(ByVal pt As Point) As Boolean

[JScript] protected override function GetHitTest(pt : Point) : Boolean;

### *Description*

Indicates whether the specified point was within the bounds of the component.  
*Return Value:* **true** if the point at the cursor should be live; otherwise, **false** .

For a form, if it has autoscroll set active, the scroll bars are always UI active. The point, in screen coordinates, of the cursor.

**r) WndProc**

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message\* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

*Description*

Provides processing for Windows messages. The message being processed.

SelectionRules enumeration (System.Windows.Forms.Design)

**a) WndProc**

*Description*

Specifies a set of selection rule identifiers that can be used to indicate attributes for a selected component.

which can be used to determine whether the selected component has some form of visible user interface and whether the user can take actions to move or alter the size of an object.

**b) WndProc**

[C#] public const SelectionRules AllSizeable;

[C++] public: const SelectionRules AllSizeable;

[VB] Public Const AllSizeable As SelectionRules

[JScript] public var AllSizeable : SelectionRules;

*Description*

Indicates the given component supports sizing in all directions, and the selection service is not currently locked.

*c) WndProc*

[C#]	public	const	SelectionRules	BottomSizeable;
[C++]	public:	const	SelectionRules	BottomSizeable;
[VB]	Public	Const	BottomSizeable	As SelectionRules
[JScript]	public	var	BottomSizeable	: SelectionRules;

*Description*

Indicates the given component supports resize from the bottom. This bit will be ignored unless the Sizeable bit is also set.

*d) WndProc*

[C#]	public	const	SelectionRules	LeftSizeable;
[C++]	public:	const	SelectionRules	LeftSizeable;
[VB]	Public	Const	LeftSizeable	As SelectionRules
[JScript]	public	var	LeftSizeable	: SelectionRules;

*Description*

Indicates the given component supports resize from the left. This bit will be ignored unless the Sizeable bit is also set.

**e) WndProc**

[C#]	public	const	SelectionRules	Locked;
[C++]	public:	const	SelectionRules	Locked;
[VB]	Public	Const	Locked	As SelectionRules
[JScript]	public	var	Locked	: SelectionRules;

*Description*

Indicates the given component is locked to its container. Overrides the moveable and sizeable properties of this enum.

**f) WndProc**

[C#]	public	const	SelectionRules	Moveable;
[C++]	public:	const	SelectionRules	Moveable;
[VB]	Public	Const	Moveable	As SelectionRules
[JScript]	public	var	Moveable	: SelectionRules;

*Description*

Indicates the given component supports a location property that allows it to be moved on the screen, and that the selection service is not currently locked.

**g) WndProc**

[C#]	public	const	SelectionRules	None;
[C++]	public:	const	SelectionRules	None;
[VB]	Public	Const	None	As SelectionRules
[JScript]	public	var	None	: SelectionRules;

*Description*

Indicates no special selection attributes.

***h) WndProc***

[C#]	public	const	SelectionRules	RightSizeable;
[C++]	public:	const	SelectionRules	RightSizeable;
[VB]	Public	Const	RightSizeable	As SelectionRules
[JScript]	public	var	RightSizeable	: SelectionRules;

*Description*

Indicates the given component supports resize from the right. This bit will be ignored unless the Sizeable bit is also set.

***i) WndProc***

[C#]	public	const	SelectionRules	TopSizeable;
[C++]	public:	const	SelectionRules	TopSizeable;
[VB]	Public	Const	TopSizeable	As SelectionRules
[JScript]	public	var	TopSizeable	: SelectionRules;

*Description*

Indicates the given component supports resize from the top. This bit will be ignored unless the Sizeable bit is also set.

j) *WndProc*

[C#]	public	const	SelectionRules	Visible;
[C++]	public:	const	SelectionRules	Visible;
[VB]	Public	Const	Visible	As SelectionRules
[JScript]	public	var	Visible	: SelectionRules;

*Description*

Indicates the given component has some form of visible user interface and the selection service is drawing a selection border around this user interface. If a selected component has this rule set, you can assume that the component implements **System.ComponentModel.IComponent** and that it is associated with a corresponding design instance.

ComponentEditorForm class (System.Windows.Forms.Design)

a) *ToString*

*Description*

Provides a user interface for a **System.Windows.Forms.Design.WindowsFormsComponentEditor** .

A **System.Windows.Forms.Design.ComponentEditorForm** shows a view of available component editor pages along with a user interface for selection.

b) *ComponentEditorForm*

*Example Syntax:*

c) *ToString*

[C#]	public	ComponentEditorForm(object component, Type[] pageTypes);
[C++]	public:	ComponentEditorForm(Object* component, Type* pageTypes[]);

1 [VB] Public Sub New(ByVal component As Object, ByVal pageTypes() As Type)  
2 [JScript] public function ComponentEditorForm(component : Object, pageTypes :  
3 Type[]);  
4

5 *Description*

6 Initializes a new instance of the  
7 **System.Windows.Forms.Design.ComponentEditorForm** class. The  
8 component to be edited. The set of  
9 **System.Windows.Forms.Design.ComponentEditorPage** objects to be  
10 shown in the form.  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25



- d) *AcceptButton*
- e) *AccessibilityObject*
- f) *AccessibleDefaultActionDescription*
- g) *AccessibleDescription*
- h) *AccessibleName*
- i) *AccessibleRole*
- j) *ActiveControl*
- k) *ActiveMdiChild*
- l) *AllowDrop*
- m) *AllowTransparency*
- n) *Anchor*
- o) *AutoScale*
- p) *AutoScaleBaseSize*
- q) *AutoScroll*
- r) *AutoScrollMargin*
- s) *AutoScrollMinSize*
- t) *AutoScrollPosition*
- u) *BackColor*
- v) *BackgroundImage*
- w) *BindingContext*
- x) *Bottom*
- y) *Bounds*
- z) *CancelButton*

1 **aa) CanFocus**  
2 **bb) CanSelect**  
3 **cc) Capture**  
4 **dd) CausesValidation**  
5 **ee) ClientRectangle**  
6 **ff) ClientSize**  
7 **gg) CompanyName**  
8 **hh) Container**  
9 **ii) ContainsFocus**  
10 **jj) ContextMenu**  
11 **kk) ControlBox**  
12 **ll) Controls**  
13 **mm) Created**  
14 **nn) CreateParams**  
15 **oo) Cursor**  
16 **pp) DataBindings**  
17 **qq) DefaultImeMode**  
18 **rr) DefaultSize**  
19 **ss) DesignMode**  
20 **tt) DesktopBounds**  
21 **uu) DesktopLocation**  
22 **vv) DialogResult**  
23 **ww) DisplayRectangle**

1	<b>xx)</b>	<b><i>Disposing</i></b>
2	<b>yy)</b>	<b><i>Dock</i></b>
3	<b>zz)</b>	<b><i>DockPadding</i></b>
4	<b>aaa)</b>	<b><i>Enabled</i></b>
5	<b>bbb)</b>	<b><i>Events</i></b>
6	<b>ccc)</b>	<b><i>Focused</i></b>
7	<b>ddd)</b>	<b><i>Font</i></b>
8	<b>eee)</b>	<b><i>FontHeight</i></b>
9	<b>fff)</b>	<b><i>ForeColor</i></b>
10	<b>ggg)</b>	<b><i>FormBorderStyle</i></b>
11	<b>hhh)</b>	<b><i>Handle</i></b>
12	<b>iii)</b>	<b><i>HasChildren</i></b>
13	<b>jjj)</b>	<b><i>Height</i></b>
14	<b>kkk)</b>	<b><i>HelpButton</i></b>
15	<b>lll)</b>	<b><i>HScroll</i></b>
16	<b>mmm)</b>	<b><i>Icon</i></b>
17	<b>nnn)</b>	<b><i>ImeMode</i></b>
18	<b>ooo)</b>	<b><i>InvokeRequired</i></b>
19	<b>ppp)</b>	<b><i>IsAccessible</i></b>
20	<b>qqq)</b>	<b><i>IsDisposed</i></b>
21	<b>rrr)</b>	<b><i>IsHandleCreated</i></b>
22	<b>sss)</b>	<b><i>IsMdiChild</i></b>
23	<b>ttt)</b>	<b><i>IsMdiContainer</i></b>
24		
25		

uuu) *IsRestrictedWindow*

vvv) *KeyPreview*

www) *Left*

xxx) *Location*

yyy) *MaximizeBox*

zzz) *MaximizedBounds*

aaaa) *MaximumSize*

bbbb) *MdiChildren*

cccc) *MdiParent*

dddd) *Menu*

eeee) *MergedMenu*

ffff) *MinimizeBox*

gggg) *MinimumSize*

hhhh) *Modal*

iiii) *Name*

jjjj) *Opacity*

kkkk) *OwnedForms*

llll) *Owner*

mmmm) *Parent*

nnnn) *ParentForm*

oooo) *ProductName*

pppp) *ProductVersion*

qqqq) *RecreatingHandle*

MSI-861US APP  
509-324-9256  
lee@hayes p16

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*rrrr) Region*  
*ssss) RenderRightToLeft*  
*tttt) ResizeRedraw*  
*uuuu) Right*  
*vvvv) RightToLeft*  
*www)ShowFocusCues*  
*xxxx) ShowInTaskbar*  
*yyyy) ShowKeyboardCues*  
*zzzz) Site*  
*aaaaa) Size*  
*bbbb) SizeGripStyle*  
*cccc) StartPosition*  
*dddd) TabIndex*  
*eeee) TabStop*  
*ffff) Tag*  
*ggggg) Text*  
*hhhhh)Top*  
*iiii) TopLevel*  
*jjjj) TopLevelControl*  
*kkkkk) TopMost*  
*llll) TransparencyKey*  
*mmmm)Visible*  
*nnnnn)VScroll*

*ooooo) Width*

*ppppp) WindowState*

*qqqqq) WindowTarget*

*rrrrr) OnActivated*

[C#] protected override void OnActivated(EventArgs e);

[C++] protected: void OnActivated(EventArgs\* e);

[VB] Overrides Protected Sub OnActivated(ByVal e As EventArgs)

[JScript] protected override function OnActivated(e : EventArgs);

*Description*

*sssss) OnHelpRequested*

[C#] protected override void OnHelpRequested(HelpEventArgs e);

[C++] protected: void OnHelpRequested(HelpEventArgs\* e);

[VB] Overrides Protected Sub OnHelpRequested(ByVal e As HelpEventArgs)

[JScript] protected override function OnHelpRequested(e : HelpEventArgs);

*Description*

*ttttt) OnSelChangeSelector*

[C#] protected virtual void OnSelChangeSelector(object source,  
TreeViewEventArgs e);

[C++] protected: virtual void OnSelChangeSelector(Object\* source,  
TreeViewEventArgs\* e);

```

1 [VB] Overridable Protected Sub OnSelChangeSelector(ByVal source As Object,
2   ByVal e As TreeViewEventArgs)
3 [JScript] protected function OnSelChangeSelector(source : Object, e :
4   TreeViewEventArgs);

```

### *Description*

Handles switching between pages.

### *uuuuu)PreProcessMessage*

```

10 [C#] public override bool PreProcessMessage(ref Message msg);
11 [C++] public: bool PreProcessMessage(Message* msg);
12 [VB] Overrides Public Function PreProcessMessage(ByRef msg As Message) As
13   Boolean
14 [JScript] public override function PreProcessMessage(msg : Message) : Boolean;

```

### *Description*

Provides a method to override in order to pre-process input messages before they are dispatched.

**System.Windows.Forms.Design.ComponentEditorForm.PreProcessMessage(System.Windows.Forms.Message@)** is called by the application's message loop to pre-process input messages before they are dispatched. A MSG structure that specifies the message to preprocess. Possible values for this parameter include WM\_KEYDOWN, WM\_SYSKEYDOWN, WM\_CHAR, and WM\_SYSCHAR.

### *vvvvv) ShowForm*

```

24 [C#] public virtual DialogResult ShowForm();

```

```

1 [C++]      public:      virtual      DialogResult      ShowForm();
2 [VB]      Overridable  Public  Function  ShowForm()  As  DialogResult
3 [JScript] public function ShowForm() : DialogResult; Shows the form. The form
4 will          have          no          owner          window.

```

#### 6 *Description*

7 Shows the form. The form will have no owner window.

8 *Return Value:* A **System.Windows.Forms.DialogResult** value indicating the result of showing the dialog.

9 *www)ShowForm*

```

10
11 [C#]      public      virtual      DialogResult      ShowForm(int      page);
12 [C++]      public:      virtual      DialogResult      ShowForm(int      page);
13 [VB]      Overridable  Public  Function  ShowForm(ByVal  page  As  Integer)  As
14 DialogResult
15 [JScript]  public  function  ShowForm(page  :  int)  :  DialogResult;
16

```

#### 17 *Description*

18 Shows the form and the specified page. The form will have no owner window.

19 *Return Value:* A **System.Windows.Forms.DialogResult** value indicating the result of showing the dialog. The index of the page to show.

20 *xxxxx)ShowForm*

```

21
22 [C#]      public      virtual      DialogResult      ShowForm(IWin32Window  owner);
23 [C++]      public:      virtual      DialogResult      ShowForm(IWin32Window*  owner);
24 [VB]      Overridable  Public  Function  ShowForm(ByVal  owner  As  IWin32Window)
25

```



As DialogResult

[JScript] public function ShowForm(owner : IWin32Window) : DialogResult;

#### *Description*

Shows the form with the specified owner.

*Return Value:* A **System.Windows.Forms.DialogResult** value indicating the result of showing the dialog. The owner of the dialog.

#### *yyyyy) ShowForm*

[C#] public virtual DialogResult ShowForm(IWin32Window owner, int page);

[C++] public: virtual DialogResult ShowForm(IWin32Window\* owner, int page);

[VB] Overridable Public Function ShowForm(ByVal owner As IWin32Window,

ByVal page As Integer) As DialogResult

[JScript] public function ShowForm(owner : IWin32Window, page : int) :

DialogResult;

#### *Description*

Shows the form and the specified page with the specified owner.

*Return Value:* A **System.Windows.Forms.DialogResult** value indicating the result of showing the dialog. The owner of the dialog. The index of the page to show.

ComponentEditorPage class (System.Windows.Forms.Design)

#### *a) WndProc*

#### *Description*

Provides a base implementation for a

**System.Windows.Forms.Design.ComponentEditorPage** .

**System.Windows.Forms.Design.ComponentEditorPage** is a complete implementation for a component editor page that consists of an empty window. You can extend this page to add your own controls.

*b) ComponentEditorPage*

*Example Syntax:*

*c) WndProc*

[C#]	public	ComponentEditorPage();
[C++]	public:	ComponentEditorPage();
[VB]	Public	Sub New()
[JScript]	public	function ComponentEditorPage();

*Description*

Initializes a new instance of the **System.Windows.Forms.Design.ComponentEditorPage** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoScroll*
- l) *AutoScrollMargin*
- m) *AutoScrollMinSize*
- n) *AutoScrollPosition*
- o) *BackColor*
- p) *BackgroundImage*
- q) *BindingContext*
- r) *BorderStyle*
- s) *Bottom*
- t) *Bounds*
- u) *CanFocus*
- v) *CanSelect*
- w) *Capture*
- x) *CausesValidation*
- y) *ClientRectangle*
- z) *ClientSize*

1        *aa)    CommitOnDeactivate*

2        *bb)    WndProc*

3  
4  
5        *Description*

6        Indicates whether an editor should apply its changes before it is deactivated.

7        The standard implementation returns **false** .

8        *cc)    CompanyName*

9        *dd)    Component*

10       *ee)    WndProc*

11  
12  
13       *Description*

14       Gets or sets the component to edit.

15       The component to edit.

- ff) Container*
- gg) ContainsFocus*
- hh) ContextMenu*
- ii) Controls*
- jj) Created*
- kk) CreateParams*
- ll) WndProc*

### *Description*

Gets or sets the creation parameters for this control.

*mm) Cursor*  
*nn) DataBindings*  
*oo) DefaultImeMode*  
*pp) DefaultSize*  
*qq) DesignMode*  
*rr) DisplayRectangle*  
*ss) Disposing*  
*tt) Dock*  
*uu) DockPadding*  
*vv) Enabled*  
*ww) Events*  
*xx) FirstActivate*  
*yy) WndProc*

### *Description*

Indicates whether the page is being activated for the first time.

zz) *Focused*

aaa) *Font*

bbb) *FontHeight*

ccc) *ForeColor*

ddd) *Handle*

eee) *HasChildren*

fff) *Height*

ggg) *HScroll*

hhh) *Icon*

iii) *WndProc*

#### *Description*

Gets or sets the icon for this page.

The caller may use this before the editor is actually activated.

*jjj) ImeMode*

*kkk) InvokeRequired*

*lll) IsAccessible*

*mmm) IsDisposed*

*nnn) IsHandleCreated*

*ooo) Left*

*ppp) Loading*

*qqq) WndProc*

#### *Description*

Indicates if loading is taking place.

*rrr) LoadRequired*

*sss) WndProc*

[C#]           protected           bool           LoadRequired           {get;           set;}

[C++] protected: \_\_property bool get\_LoadRequired();protected: \_\_property void  
set\_LoadRequired(bool);

[VB]           Protected           Property           LoadRequired           As           Boolean

[JScript] protected function get LoadRequired() : Boolean;protected function set  
LoadRequired(Boolean);

#### *Description*

Indicates whether a load is required previous to editing.



1        *ttt)    Location*

2        *uuu)    Name*

3        *vvv)    PageSite*

4        *www)   WndProc*

5  
6  
7        *Description*

8        Gets or sets the page site.

9        The site for this component editor page.

xxx)	<i>Parent</i>
yyy)	<i>ProductName</i>
zzz)	<i>ProductVersion</i>
aaaa)	<i>RecreatingHandle</i>
bbbb)	<i>Region</i>
cccc)	<i>RenderRightToLeft</i>
dddd)	<i>ResizeRedraw</i>
eeee)	<i>Right</i>
ffff)	<i>RightToLeft</i>
gggg)	<i>ShowFocusCues</i>
hhhh)	<i>ShowKeyboardCues</i>
iiii)	<i>Site</i>
jjjj)	<i>Size</i>
kkkk)	<i>TabIndex</i>
llll)	<i>TabStop</i>
mmmm)	<i>Tag</i>
nnnn)	<i>Text</i>
oooo)	<i>Title</i>
pppp)	<i>WndProc</i>

#### *Description*

Gets or sets the title of the page.

The caller may use this to show the name of this editor before it's actually activated.

*qqqq) Top*

*rrrr) TopLevelControl*

*ssss) Visible*

*tttt) VScroll*

*uuuu) Width*

*vvvv) WindowTarget*

*www)Activate*

[C#]	public	virtual	void	Activate();
[C++]	public:	virtual	void	Activate();
[VB]	Overridable	Public	Sub	Activate()
[JScript]	public	function		Activate();

#### *Description*

Activates and displays the page.

*xxxx) ApplyChanges*

[C#]	public	virtual	void	ApplyChanges();
[C++]	public:	virtual	void	ApplyChanges();
[VB]	Overridable	Public	Sub	ApplyChanges()
[JScript]	public	function		ApplyChanges();

#### *Description*

Applies changes to all the components being edited.

*yyyy) Deactivate*

[C#]	public	virtual	void	Deactivate();
[C++]	public:	virtual	void	Deactivate();
[VB]	Overridable	Public	Sub	Deactivate()
[JScript]	public	function		Deactivate();

*Description*

Deactivates and hides the page.

*zzzz) EnterLoadingMode*

[C#]	protected	void	EnterLoadingMode();
[C++]	protected:	void	EnterLoadingMode();
[VB]	Protected	Sub	EnterLoadingMode()
[JScript]	protected	function	EnterLoadingMode();

*Description*

Increments the loading counter, which determines whether a page is in loading mode.

*aaaaa) ExitLoadingMode*

[C#]	protected	void	ExitLoadingMode();
[C++]	protected:	void	ExitLoadingMode();
[VB]	Protected	Sub	ExitLoadingMode()

```
[JScript]           protected           function           ExitLoadingMode();
```

### *Description*

Decrements the loading counter, which determines whether a page is in loading mode.

### *bbbbbb) GetControl*

```
[C#]           public           virtual           Control           GetControl();
```

```
[C++]           public:           virtual           Control*           GetControl();
```

```
[VB]   Overridable   Public   Function   GetControl()   As   Control
```

```
[JScript]       public           function           GetControl()       :       Control;
```

### *Description*

Gets the control that represents the window for this page.

### *cccccc) GetSelectedComponent*

```
[C#]           protected           IComponent           GetSelectedComponent();
```

```
[C++]           protected:           IComponent*           GetSelectedComponent();
```

```
[VB]   Protected   Function   GetSelectedComponent()   As   IComponent
```

```
[JScript]       protected   function   GetSelectedComponent()   :   IComponent;
```

### *Description*

Gets the component that is to be edited.

*Return Value:* The component that is being edited.

### *dddd) IsFirstActivate*

```
[C#]          protected          bool          IsFirstActivate();
[C++]          protected:          bool          IsFirstActivate();
[VB]    Protected    Function    IsFirstActivate()    As    Boolean
[JScript]    protected    function    IsFirstActivate()    :    Boolean;
```

#### *Description*

Gets a value indicating whether the page is being activated for the first time.

*Return Value:* **true** if this is the first time the page is being activated; otherwise, **false** .

### *eeee) IsLoading*

```
[C#]          protected          bool          IsLoading();
[C++]          protected:          bool          IsLoading();
[VB]    Protected    Function    IsLoading()    As    Boolean
[JScript]    protected    function    IsLoading()    :    Boolean;
```

#### *Description*

Gets a value indicating whether the page is being loaded.

*Return Value:* **true** if this is the page is being loaded; otherwise, **false** .

### *ffff) IsPageMessage*

```
[C#]    public    virtual    bool    IsPageMessage(ref    Message    msg);
[C++]    public:    virtual    bool    IsPageMessage(Message*    msg);
[VB]    Overridable Public Function IsPageMessage(ByRef msg As Message) As
```

Boolean

[JScript] public function IsPageMessage(msg : Message) : Boolean;

#### *Description*

Processes messages that could be handled by the page.

*Return Value:* **true** if the page processed the message; otherwise, **false** .

Gives the page a chance to process messages before the caller uses them. If this returns **true** , the message will be eaten. Otherwise, the caller will continue to process the message. Message to process.

#### *ggggg) LoadComponent*

[C#] protected abstract void LoadComponent();

[C++] protected: virtual void LoadComponent() = 0;

[VB] MustOverride Protected Sub LoadComponent()

[JScript] protected abstract function LoadComponent();

#### *Description*

Loads the component into the page UI.

Each page must override this and implement its own custom behavior.

#### *hhhhh)OnApplyComplete*

[C#] public virtual void OnApplyComplete();

[C++] public: virtual void OnApplyComplete();

[VB] Overridable Public Sub OnApplyComplete()

[JScript] public function OnApplyComplete();

## Description

Called when the page along with its sibling pages have applied their changes.

**System.Windows.Forms.Design.ComponentEditorPage** gives a chance for the page to reload itself especially when they are interdependent, and need to reflect changes made in another page.

### iiii) ReloadComponent

[C#]	protected	virtual	void	ReloadComponent();
[C++]	protected:	virtual	void	ReloadComponent();
[VB]	Overridable	Protected	Sub	ReloadComponent()
[JScript]	protected	function		ReloadComponent();

## Description

Called when the current component may have changed elsewhere and needs to be reloaded into the UI.

### jjjj) SaveComponent

[C#]	protected	abstract	void	SaveComponent();
[C++]	protected:	virtual	void	SaveComponent() = 0;
[VB]	MustOverride	Protected	Sub	SaveComponent()
[JScript]	protected	abstract	function	SaveComponent();

## Description

Saves the component from the page UI.

Each page must override this and implement its own custom behavior.



### *kkkkk) SetComponent*

```
[C#]    public    virtual    void    SetComponent(IComponent    component);
[C++]   public:   virtual    void    SetComponent(IComponent*    component);
[VB]    Overridable Public Sub SetComponent(ByVal component As IComponent)
[JScript] public    function    SetComponent(component    :    IComponent);
```

#### *Description*

Sets the component to be edited. The component to be edited.

### *lllll) SetDirty*

```
[C#]          protected          virtual          void          SetDirty();
[C++]         protected:         virtual          void          SetDirty();
[VB]          Overridable         Protected        Sub          SetDirty()
[JScript]          protected          function          SetDirty();
```

#### *Description*

Sets the page to be in dirty state.

### *mmmmm) SetSite*

```
[C#]    public    virtual    void    SetSite(IComponentEditorPageSite    site);
[C++]   public:   virtual    void    SetSite(IComponentEditorPageSite*    site);
[VB]    Overridable Public Sub SetSite(ByVal site As IComponentEditorPageSite)
[JScript] public    function    SetSite(site    :    IComponentEditorPageSite);
```

## Description

Sets the site for this page. Site for this page.

### *nnnnn)ShowHelp*

[C#]	public	virtual	void	ShowHelp();
[C++]	public:	virtual	void	ShowHelp();
[VB]	Overridable	Public	Sub	ShowHelp()
[JScript]	public	function	ShowHelp();	

## Description

Provides help information to the help system.

This is only called if

**System.Windows.Forms.Design.ComponentEditorPage.SupportsHelp** returns **true** . The help system calls **System.Windows.Forms.Design.ComponentEditorPage.ShowHelp** to provide help for the page.

### *ooooo) SupportsHelp*

[C#]	public	virtual	bool	SupportsHelp();
[C++]	public:	virtual	bool	SupportsHelp();
[VB]	Overridable	Public	Function	SupportsHelp() As Boolean
[JScript]	public	function	SupportsHelp()	: Boolean;

## Description

Gets a value indicating whether the editor supports Help.

**Return Value:** **true** if the editor supports help; otherwise, **false** . The default implementation returns **false** .

EventsTab class (System.Windows.Forms.Design)

*a) WndProc*

*Description*

Provides a **System.Windows.Forms.Design.PropertyTab** which can display events for selection and linking.

*b) EventsTab*

*Example Syntax:*

*c) WndProc*

[C#]            public            EventsTab(IServiceProvider            sp);

[C++]            public:            EventsTab(IServiceProvider\*            sp);

[VB]    Public    Sub    New(ByVal    sp    As    IServiceProvider)

[JScript]    public    function    EventsTab(sp    :    IServiceProvider);

*Description*

Initializes a new instance of the **System.Windows.Forms.Design.EventsTab** class. A service object provider to use.

- d) *Bitmap*
- e) *Components*
- f) *HelpKeyword*
- g) *WndProc*

#### *Description*

Gets or sets the help keyword for the tab.

- h) *TabName*
- i) *WndProc*

```
[C#]      public      override      string      TabName      {get;}
[C++]      public:      __property      virtual      String*      get_TabName();
[VB]      Overrides      Public      ReadOnly      Property      TabName      As      String
[JScript]      public      function      get      TabName()      :      String;
```

#### *Description*

Gets or sets the name of the tab.

- j) *CanExtend*

```
[C#]      public      override      bool      CanExtend(object      extendee);
[C++]      public:      bool      CanExtend(Object*      extendee);
[VB]      Overrides      Public      Function      CanExtend(ByVal      extendee      As      Object)      As
Boolean
[JScript]      public      override      function      CanExtend(extendee : Object) : Boolean;
```

## *Description*

Gets a value indicating whether the specified object can be extended.

*Return Value:* **true** if the specified object may be extended; otherwise, **false**.

The object to determine whether can be extended.

### *k) GetDefaultProperty*

[C#] public override PropertyDescriptor GetDefaultProperty(object obj);

[C++] public: PropertyDescriptor\* GetDefaultProperty(Object\* obj);

[VB] Overrides Public Function GetDefaultProperty(ByVal obj As Object) As PropertyDescriptor

[JScript] public override function GetDefaultProperty(obj : Object) : PropertyDescriptor;

## *Description*

Gets the default property from the specified object.

*Return Value:* A **System.ComponentModel.PropertyDescriptor** indicating the default property. The object to retrieve the default property of.

### *l) GetProperties*

[C#] public override PropertyDescriptorCollection GetProperties(object component, Attribute[] attributes);

[C++] public: PropertyDescriptorCollection\* GetProperties(Object\* component, Attribute\* attributes[]);

[VB] Overrides Public Function GetProperties(ByVal component As Object, ByVal attributes() As Attribute) As PropertyDescriptorCollection

```

1 [JScript] public override function GetProperties(component : Object, attributes :
2 Attribute[]) : PropertyDescriptorCollection;

```

#### 4 *Description*

5 Gets all the properties of the event tab that match the specified attributes. The  
6 event tab properties are determined from the event properties returned from a  
7 component's event service.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection**  
8 that indicates the properties. This will be a default initialized  
9 **System.ComponentModel.PropertyDescriptorCollection** if the component  
10 does not implement an event service. The component to retrieve the properties  
11 of. An array of **System.Attribute** that indicates the attributes of the event  
12 properties to retrieve.

#### 10 *m) GetProperties*

```

12 [C#] public override PropertyDescriptorCollection
13 GetProperties(ITypeDescriptorContext context, object component, Attribute[]
14 attributes);

```

```

15 [C++] public: PropertyDescriptorCollection*
16 GetProperties(ITypeDescriptorContext* context, Object* component, Attribute*
17 attributes[]);

```

```

18 [VB] Overrides Public Function GetProperties(ByVal context As
19 ITypeDescriptorContext, ByVal component As Object, ByVal attributes() As
20 Attribute) As PropertyDescriptorCollection

```

```

21 [JScript] public override function GetProperties(context : ITypeDescriptorContext,
22 component : Object, attributes : Attribute[]) : PropertyDescriptorCollection; Gets
23 the properties of the specified component...

```

IUIService interface (System.Windows.Forms.Design)

*a) ToString*

*Description*

Provides support for interaction with the user interface of the development environment object that is hosting the designer.

**System.Windows.Forms.Design.IUIService** can display error messages, show dialog boxes, and get ambient properties of the host, such as the font for dialogs and color schemes, through the Styles dictionary property.

*b) Styles*

*c) ToString*

[C#]	IDictionary	Styles	{get;}
[C++]	IDictionary*		get_Styles();
[VB]	ReadOnly Property	Styles As	IDictionary
[JScript]	abstract function	get Styles() :	IDictionary;

*Description*

Gets or sets the collections of styles that are specific to the host's environment.

The dictionary can provide information from the host environment. At a minimum, this consists of the font that should be used for standard UI text, and the color to use for highlighting. These required styles are "DialogFont" and "HighlightColor".

*d) CanShowComponentEditor*

[C#]	bool	CanShowComponentEditor(object	component);
------	------	-------------------------------	-------------

```

1  [C++]      bool      CanShowComponentEditor(Object*      component);
2  [VB] Function CanShowComponentEditor(ByVal component As Object) As
3  Boolean
4  [JScript] function CanShowComponentEditor(component : Object) : Boolean;
5

```

#### *Description*

Indicates whether the component can display a  
**System.ComponentModel.Design.ComponentDesigner** .

Component editors are similar to property pages from COM. The component to  
check for support for displaying a  
**System.ComponentModel.Design.ComponentDesigner**.

#### *e) GetDialogOwnerWindow*

```

12 [C#]          IWin32Window      GetDialogOwnerWindow();
13 [C++]          IWin32Window*      GetDialogOwnerWindow();
14 [VB] Function  GetDialogOwnerWindow() As IWin32Window
15 [JScript] function  GetDialogOwnerWindow() : IWin32Window;
16

```

#### *Description*

Gets the window that should be used for dialog parenting.

#### *f) SetUIDirty*

```

21 [C#]          void      SetUIDirty();
22 [C++]          void      SetUIDirty();
23 [VB] Sub      SetUIDirty()
24 [JScript] function      SetUIDirty();
25

```



## Description

Sets a flag indicating the UI is dirty.

The UI becomes dirty whenever a toolbar or menu item's status changes. Most development environments cache the status of these elements for speed, and need to know when they need to be updated. This method would be called, for example, after a set of objects have been selected within the designer to enable the cut and copy menu items, for example.

### *g) ShowComponentEditor*

[C#] bool ShowComponentEditor(object component, IWin32Window parent);

[C++] bool ShowComponentEditor(Object\* component, IWin32Window\* parent);

[VB] Function ShowComponentEditor(ByVal component As Object, ByVal parent As IWin32Window) As Boolean

[JScript] function ShowComponentEditor(component : Object, parent : IWin32Window) : Boolean;

## Description

Attempts to display a

**System.ComponentModel.Design.ComponentDesigner** for a component.

*Return Value:* **true** if the attempt was successful; otherwise, **false**.

A **System.ComponentModel.Design.ComponentDesigner** is similar to a property page from COM. The component for which to display a **System.ComponentModel.Design.ComponentDesigner**. The object to parent any dialogs to.

### *h) ShowDialog*

[C#] DialogResult ShowDialog(Form form);

```

1  [C++]          DialogResult          ShowDialog(Form*          form);
2  [VB]  Function  ShowDialog(ByVal  form  As  Form)  As  DialogResult
3  [JScript]  function  ShowDialog(form  :  Form)  :  DialogResult;

```

#### *Description*

Attempts to display the specified form in a dialog box.

*Return Value:* A **System.Windows.Forms.DialogResult** indicating the results of the dialog box.

#### *i) ShowError*

```

10 [C#]          void          ShowError(Exception          ex);
11 [C++]          void          ShowError(Exception*          ex);
12 [VB]  Sub      ShowError(ByVal      ex      As      Exception)
13 [JScript]      function      ShowError(ex      :      Exception);

```

#### *Description*

Displays the specified exception and its information in a message box.

Using this method allows the message box display to be properly integrated with the development environment. The exception to display.

#### *j) ShowError*

```

20 [C#]          void          ShowError(string          message);
21 [C++]          void          ShowError(String*          message);
22 [VB]  Sub      ShowError(ByVal      message      As      String)
23 [JScript]  function  ShowError(message : String); Displays the specified error
24 message          in          a          message          box.
25

```

## Description

Displays the specified error message in a message box.

Using this method allows the message box display to be properly integrated with the development environment. The error message to display.

### k) *ShowError*

[C#] void ShowError(Exception ex, string message);

[C++] void ShowError(Exception\* ex, String\* message);

[VB] Sub ShowError(ByVal ex As Exception, ByVal message As String)

[JScript] function ShowError(ex : Exception, message : String);

## Description

Displays the specified exception and its information in a message box.

Using this method allows the message box display to be properly integrated with the development environment. The exception to display. A message to display that provides information about the exception.

### l) *ShowMessage*

[C#] void ShowMessage(string message);

[C++] void ShowMessage(String\* message);

[VB] Sub ShowMessage(ByVal message As String)

[JScript] function ShowMessage(message : String); Displays the specified message in a message box.

## Description

Displays the specified message in a message box.

Using this method allows the message box display to be properly integrated with the development environment. The message to display

*m) ShowMessage*

[C#] void ShowMessage(string message, string caption);

[C++] void ShowMessage(String\* message, String\* caption);

[VB] Sub ShowMessage(ByVal message As String, ByVal caption As String)

[JScript] function ShowMessage(message : String, caption : String);

*Description*

Displays the specified message in a message box with the specified caption.

Using this method allows the message box display to be properly integrated with the development environment. The message to display. The caption for the message box.

*n) ShowMessage*

[C#] DialogResult ShowMessage(string message, string caption, MessageBoxButtons buttons);

[C++] DialogResult ShowMessage(String\* message, String\* caption, MessageBoxButtons buttons);

[VB] Function ShowMessage(ByVal message As String, ByVal caption As String, ByVal buttons As MessageBoxButtons) As DialogResult

[JScript] function ShowMessage(message : String, caption : String, buttons : MessageBoxButtons) : DialogResult;

## Description

Displays the specified message in a message box with the specified caption and buttons to place on the dialog box.

*Return Value:* A **System.Windows.Forms.DialogResult** indicating the results of the dialog box.

Using this method allows the message box display to be properly integrated with the development environment. The message to display. The caption for the dialog box. The buttons to place on the dialog box. Must be one of the following values defined in **System.Windows.Forms.MessageBoxButtons** :

**System.Windows.Forms.MessageBoxButtons.OK,**  
**System.Windows.Forms.MessageBoxButtons.OKCancel,**  
**System.Windows.Forms.MessageBoxButtons.YesNo,**  
**System.Windows.Forms.MessageBoxButtons.YesNoCancel.**

### *o) ShowToolWindow*

[C#]	bool	ShowToolWindow(Guid	toolWindow);
[C++]	bool	ShowToolWindow(Guid	toolWindow);
[VB]	Function	ShowToolWindow(ByVal toolWindow As Guid)	As Boolean
[JScript]	function	ShowToolWindow(toolWindow : Guid)	: Boolean;

## Description

Displays the specified tool window.

*Return Value:* **true** if the tool window was successfully shown; **false** if it couldn't be shown or found. An identifier for the tool window. This can be a custom Guid or one of the predefined values from

**System.ComponentModel.Design.StandardToolWindows.**

IWindowsFormsEditorService interface (System.Windows.Forms.Design)

**a) ShowToolWindow**

**b) CloseDropDown**

[C#] void CloseDropDown();

[C++] void CloseDropDown();

[VB] Sub CloseDropDown()

[JScript] function CloseDropDown(); Closes a previously opened drop down list.

**c) DropDownControl**

[C#] void DropDownControl(Control control);

[C++] void DropDownControl(Control\* control);

[VB] Sub DropDownControl(ByVal control As Control)

[JScript] function DropDownControl(control : Control); Displays the specified control in a drop down list.

**d) ShowDialog**

[C#] DialogResult ShowDialog(Form dialog);

[C++] DialogResult ShowDialog(Form\* dialog);

[VB] Function ShowDialog(ByVal dialog As Form) As DialogResult

[JScript] function ShowDialog(dialog : Form) : DialogResult; Shows the specified dialog box.

PropertyTab class (System.Windows.Forms.Design)

**a) ShowDialog**

*Description*

Provides a base class for property tabs.

**System.Windows.Forms.Design.PropertyTab** provides the basic functionality for a property tab which can provide an interface to properties which it creates extender providers for.

**b) PropertyTab**

*Example Syntax:*

**c) ShowDialog**

[C#]                      protected                      PropertyTab();

[C++]                      protected:                      PropertyTab();

[VB]                      Protected                      Sub                      New()

[JScript] protected function PropertyTab();

**d) Bitmap**

**e) ShowDialog**

[C#]              public              virtual              Bitmap              Bitmap              {get;}

[C++]              public:              \_\_property              virtual              Bitmap\*              get\_Bitmap();

[VB]              Overridable              Public              ReadOnly              Property              Bitmap              As              Bitmap

[JScript]              public              function              get              Bitmap()              :              Bitmap;

*Description*

Gets or sets a bitmap to display in the property tab.

*f) Components*

*g) ShowDialog*

[C#] public virtual object[] Components {get; set;}

[C++] public: \_\_property virtual Object\* get\_Components();public: \_\_property

virtual void set\_Components(Object\* \_\_gc[]);

[VB] Overridable Public Property Components As Object ()

[JScript] public function get Components() : Object[];public function set  
Components(Object[]);

*Description*

Gets or sets the array of components the property tab is associated with.

*h) HelpKeyword*

*i) ShowDialog*

[C#] public virtual string HelpKeyword {get;}

[C++] public: \_\_property virtual String\* get\_HelpKeyword();

[VB] Overridable Public ReadOnly Property HelpKeyword As String

[JScript] public function get HelpKeyword() : String;

*Description*

Gets or sets the help keyword that is to be associated with this tab.



By default, this is set to the tab name.

j) *TabName*

k) *ShowDialog*

[C#]        public        abstract        string        TabName        {get;}

[C++]    public:    \_\_property    virtual    String\*    get\_TabName()    =    0;

[VB]   MustOverride   Public   ReadOnly   Property   TabName   As   String

[JScript]   public   abstract   function   get   TabName()   :   String;

#### *Description*

Gets or sets the name for the property tab.

l) *CanExtend*

[C#]        public        virtual        bool        CanExtend(object        extendee);

[C++]    public:        virtual        bool        CanExtend(Object\*        extendee);

[VB]   Overridable   Public   Function   CanExtend(ByVal   extendee   As   Object)   As  
Boolean

[JScript]   public   function   CanExtend(extendee   :   Object)   :   Boolean;

#### *Description*

Gets a value indicating whether the specified object be can extended.

**Return Value:** **true** if the object can be extended; otherwise, **false** . The object to determine whether can be extended.

m) *Dispose*

[C#]        public        virtual        void        Dispose();

[C++]	public:	virtual	void	Dispose();
[VB]	Overridable	Public	Sub	Dispose()
[JScript]	public	function		Dispose();

## Description

### n) *Dispose*

[C#]	protected	virtual	void	Dispose(bool disposing);
[C++]	protected:	virtual	void	Dispose(bool disposing);
[VB]	Overridable	Protected	Sub	Dispose(ByVal disposing As Boolean)
[JScript]	protected	function		Dispose(disposing : Boolean);

### o) *Finalize*

[C#]				~PropertyTab();
[C++]				~PropertyTab();
[VB]	Overrides	Protected	Sub	Finalize()
[JScript]	protected	override	function	Finalize();

### p) *GetDefaultProperty*

[C#]	public	virtual	PropertyDescriptor	GetDefaultProperty(object component);
[C++]	public:	virtual	PropertyDescriptor*	GetDefaultProperty(Object* component);
[VB]	Overridable	Public	Function	GetDefaultProperty(ByVal component As

```

1      Object)                                As                                PropertyDescriptor
2      [JScript] public function GetDefaultProperty(component : Object) :
3      PropertyDescriptor;

```

#### 5 *Description*

Gets the default property of the specified component.

*Return Value:* A **System.ComponentModel.PropertyDescriptor** that represents the default property. The component to retrieve the default property of.

#### 9 *q) GetProperties*

```

10     [C#] public virtual PropertyDescriptorCollection GetProperties(object
11     component);

```

```

12     [C++] public: virtual PropertyDescriptorCollection* GetProperties(Object*
13     component);

```

```

14     [VB] Overridable Public Function GetProperties(ByVal component As Object) As
15     PropertyDescriptorCollection

```

```

16     [JScript] public function GetProperties(component : Object) :
17     PropertyDescriptorCollection; Gets the properties of the specified component.

```

#### 20 *Description*

Gets the properties of the specified component.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** containing the properties of the specified component. The component to retrieve the properties of.

r) *GetProperties*

```
[C#] public abstract PropertyDescriptorCollection GetProperties(object
component, Attribute[] attributes);
[C++] public: virtual PropertyDescriptorCollection* GetProperties(Object*
component, Attribute* attributes[]) = 0;
[VB] MustOverride Public Function GetProperties(ByVal component As Object,
ByVal attributes() As Attribute) As PropertyDescriptorCollection
[JScript] public abstract function GetProperties(component : Object, attributes :
Attribute[]) : PropertyDescriptorCollection;
```

*Description*

Gets the properties of the specified component which match the specified attributes.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** that indicates the properties. The component to retrieve properties from. An array of **System.Attribute** that indicates the attributes of the properties to retrieve.

s) *GetProperties*

```
[C#] public virtual PropertyDescriptorCollection
GetProperties(ITypeDescriptorContext context, object component, Attribute[]
attributes);
[C++] public: virtual PropertyDescriptorCollection*
GetProperties(ITypeDescriptorContext* context, Object* component, Attribute*
attributes[]);
[VB] Overridable Public Function GetProperties(ByVal context As
```

```

1 ITypeDescriptorContext, ByVal component As Object, ByVal attributes() As
2 Attribute) As PropertyDescriptorCollection
3 [JScript] public function GetProperties(context : ITypeDescriptorContext,
4 component : Object, attributes : Attribute[]) : PropertyDescriptorCollection;
5

```

#### *Description*

Gets the properties of the specified context which match the specified attributes.  
*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** that indicates the properties matching the specified context and attributes. An ITypeDescriptorContext that indicates the context to retrieve properties from. The component to retrieve properties from. An array of **System.Attribute** that indicates the attributes of the properties to retrieve.

WindowsFormsComponentEditor class (System.Windows.Forms.Design)

#### *a) ToString*

#### *Description*

Provides a base class for editors that use a modal dialog to display a properties page similar to an ActiveX control's property page.

#### *b) WindowsFormsComponentEditor*

#### *Example Syntax:*

#### *c) ToString*

```

21 [C#] protected WindowsFormsComponentEditor();
22 [C++] protected: WindowsFormsComponentEditor();
23 [VB] Protected Sub New()
24 [JScript] protected function WindowsFormsComponentEditor();
25

```

d) *EditComponent*

[C#] public override bool EditComponent(ITypeDescriptorContext context, object component);

[C++] public: bool EditComponent(ITypeDescriptorContext\* context, Object\* component);

[VB] Overrides Public Function EditComponent(ByVal context As ITypeDescriptorContext, ByVal component As Object) As Boolean

[JScript] public override function EditComponent(context : ITypeDescriptorContext, component : Object) : Boolean;

*Description*

Activates a UI used to edit the component.

e) *EditComponent*

[C#] public bool EditComponent(object component, IWin32Window owner);

[C++] public: bool EditComponent(Object\* component, IWin32Window\* owner);

[VB] Public Function EditComponent(ByVal component As Object, ByVal owner As IWin32Window) As Boolean

[JScript] public function EditComponent(component : Object, owner : IWin32Window) : Boolean;

*Description*

Activates the advanced UI used to edit the component.

*f) EditComponent*

```
[C#] public virtual bool EditComponent(ITypeDescriptorContext context, object
component, IWin32Window owner);
[C++] public: virtual bool EditComponent(ITypeDescriptorContext* context,
Object* component, IWin32Window* owner);
[VB] Overridable Public Function EditComponent(ByVal context As
ITypeDescriptorContext, ByVal component As Object, ByVal owner As
IWin32Window) As Boolean
[JScript] public function EditComponent(context : ITypeDescriptorContext,
component : Object, owner : IWin32Window) : Boolean;
```

*Description*

Activates the advanced UI used to edit the component.

*g) GetComponentEditorPages*

```
[C#] protected virtual Type[] GetComponentEditorPages();
[C++] protected: virtual Type* GetComponentEditorPages() [];
[VB] Overridable Protected Function GetComponentEditorPages() As Type()
[JScript] protected function GetComponentEditorPages() : Type[];
```

*Description*

Gets the set of **System.Windows.Forms.Design.ComponentEditorPage** pages to be used.

## h) *GetInitialComponentEditorPageIndex*

[C#] protected virtua

### 2. System.Windows.Forms

The namespace contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system. In this namespace you will find the class and many other controls that can be added to forms to create user interfaces.

#### *Description*

The **System.Windows.Forms** namespace contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system. In this namespace you will find the **System.Windows.Forms.Form** class and many other controls that can be added to forms to create user interfaces.

AxHost.AboutBoxDelegate delegate (System.Windows.Forms)

#### *Description*

Represents the method that will display an ActiveX control's about dialog box.

When you create an **System.Windows.Forms.AxHost.AboutBoxDelegate** delegate, you identify the method that will handle display an ActiveX control's about dialog box if it has one. To associate the method with your handler, add an instance of the delegate to the method. The handler is called whenever the method is called, unless you remove the delegate. For more information about delegates, see .



## AccessibleEvents enumeration (System.Windows.Forms)

### *Description*

Specifies events that are reported by accessible applications.

Accessibility events are generated by the operating system in response to changes in the user interface.

[C#]	public	const	AccessibleEvents	AcceleratorChange;
[C++]	public:	const	AccessibleEvents	AcceleratorChange;
[VB]	Public	Const	AcceleratorChange	As AccessibleEvents
[JScript]	public	var	AcceleratorChange	: AccessibleEvents;

### *Description*

An object's **System.Windows.Forms.AccessibleObject.KeyboardShortcut** property changed. Server applications send this event for their accessible objects.

[C#]	public	const	AccessibleEvents	Create;
[C++]	public:	const	AccessibleEvents	Create;
[VB]	Public	Const	Create	As AccessibleEvents
[JScript]	public	var	Create	: AccessibleEvents;

### *Description*

An object was created. The system sends this event for the following user interface elements: caret, header control, list view control, tab control, toolbar control, tree view control, and window object. Server applications send this event for their accessible objects. Servers must send this event for all of an object's

child objects before sending the event for the parent object. Servers must ensure that all child objects are fully created and ready to accept calls from clients when the parent object sends this event.

```
[C#]      public      const      AccessibleEvents      DefaultActionChange;
[C++]     public:     const      AccessibleEvents      DefaultActionChange;
[VB]      Public      Const      DefaultActionChange      As      AccessibleEvents
[JScript]  public     var      DefaultActionChange      :      AccessibleEvents;
```

#### *Description*

An object's **System.Windows.Forms.AccessibleObject.DefaultAction** property changed. The system sends this event for dialog boxes. Server applications send this event for their accessible objects. Therefore, server applications do not need to send this event for the child objects. Hidden objects have a state of **System.Windows.Forms.AccessibleStates.Invisible** and shown objects do not. Events of type **AccessibleEvents.Hide** indicate that a state of **System.Windows.Forms.AccessibleStates.Invisible** has been set. Therefore, servers do not need to send the **AccessibleEvents.StateChange** event in this case.

```
[C#]      public      const      AccessibleEvents      DescriptionChange;
[C++]     public:     const      AccessibleEvents      DescriptionChange;
[VB]      Public      Const      DescriptionChange      As      AccessibleEvents
[JScript]  public     var      DescriptionChange      :      AccessibleEvents;
```

#### *Description*

An object's **System.Windows.Forms.AccessibleObject.Description** property changed. Server applications send this event for their accessible objects.

[C#]	public	const	AccessibleEvents	Destroy;
[C++]	public:	const	AccessibleEvents	Destroy;
[VB]	Public	Const	Destroy	As AccessibleEvents
[JScript]	public	var	Destroy	: AccessibleEvents;

#### Description

An object was destroyed. The system sends this event for the following user interface elements: caret, header control, list view control, tab control, toolbar control, tree view control, and window object. Server applications send this event for their accessible objects. This event may or may not be sent for child objects. However, clients can assume that all the children of an object have been destroyed when the parent object sends this event.

[C#]	public	const	AccessibleEvents	Focus;
[C++]	public:	const	AccessibleEvents	Focus;
[VB]	Public	Const	Focus	As AccessibleEvents
[JScript]	public	var	Focus	: AccessibleEvents;

#### Description

An object has received the keyboard focus. The system sends this event for the following user interface elements: list view control, menu bar, pop-up menu, switch window, tab control, tree view control, and window object. Server applications send this event for their accessible objects.

[C#]	public	const	AccessibleEvents	HelpChange;
[C++]	public:	const	AccessibleEvents	HelpChange;
[VB]	Public	Const	HelpChange	As AccessibleEvents

1 [JScript] public var HelpChange : AccessibleEvents;

3 *Description*

4 An object's Help property changed. Server applications send this event for their  
5 accessible objects.

6 [C#] public const AccessibleEvents Hide;

7 [C++] public: const AccessibleEvents Hide;

8 [VB] Public Const Hide As AccessibleEvents

9 [JScript] public var Hide : AccessibleEvents;

11 *Description*

12 An object is being hidden. The system sends this event for the following user  
13 interface elements: caret and cursor. Server applications send this event for their  
14 accessible objects. When this event is generated for a parent object, all child  
15 objects have already been hidden. Therefore, server applications do not need to  
16 send this event for the child objects. This event is not sent consistently by the  
17 system. This is a known problem and is being addressed by the accessibility  
18 team.

18 [C#] public const AccessibleEvents LocationChange;

19 [C++] public: const AccessibleEvents LocationChange;

20 [VB] Public Const LocationChange As AccessibleEvents

21 [JScript] public var LocationChange : AccessibleEvents;

23 *Description*

24 An object has changed location, shape, or size. The system sends this event for  
25 the following user interface elements: caret and window object. Server  
applications send this event for their accessible objects. This event is generated

in response to the top-level object within the object hierarchy that has changed, not for any children it might contain. For example, if the user resizes a window, the system sends this notification for the window, but not for the menu bar, title bar, scroll bars, or other objects that have also changed. The system does not send this event for every non-floating child window when the parent moves. However, if an application explicitly resizes child windows as a result of being resized itself, the system will send multiple events for the resized children. If an object's **System.Windows.Forms.AccessibleObject.State** property is set to **System.Windows.Forms.AccessibleStates.Floating**, servers should send a location change event whenever the object changes location. If an object does not have this state, servers should raise this event when the object moves relative to its parent.

[C#]	public	const	AccessibleEvents		NameChange;
[C++]	public:	const	AccessibleEvents		NameChange;
[VB]	Public	Const	NameChange	As	AccessibleEvents
[JScript]	public	var	NameChange	:	AccessibleEvents;

#### *Description*

An object's **System.Windows.Forms.AccessibleObject.Name** property changed. The system sends this event for the following user interface elements: check box, cursor, list view control, push button, radio button, status bar control, tree view control, and window object. Server applications send this event for their accessible objects.

[C#]	public	const	AccessibleEvents		ParentChange;
[C++]	public:	const	AccessibleEvents		ParentChange;
[VB]	Public	Const	ParentChange	As	AccessibleEvents
[JScript]	public	var	ParentChange	:	AccessibleEvents;

#### *Description*

An object has a new parent object. Server applications send this event for their accessible objects.

[C#]	public	const	AccessibleEvents	Reorder;
[C++]	public:	const	AccessibleEvents	Reorder;
[VB]	Public	Const	Reorder	As AccessibleEvents
[JScript]	public	var	Reorder	: AccessibleEvents;

#### *Description*

A container object has added, removed, or reordered its children. The system sends this event for the following user interface elements: header control, list view control, toolbar control, and window object. Server applications send this event as appropriate for their accessible objects. This event is also sent by a parent window when the z order for the child windows has changed.

[C#]	public	const	AccessibleEvents	Selection;
[C++]	public:	const	AccessibleEvents	Selection;
[VB]	Public	Const	Selection	As AccessibleEvents
[JScript]	public	var	Selection	: AccessibleEvents;

#### *Description*

An accessible object within a container object has been selected. This event signals a single selection - either a child has been selected in a container that previously did not contain any selected children or the selection has changed from one child to another.

[C#]	public	const	AccessibleEvents	SelectionAdd;
[C++]	public:	const	AccessibleEvents	SelectionAdd;
[VB]	Public	Const	SelectionAdd	As AccessibleEvents

[JScript]      public      var      SelectionAdd      :      AccessibleEvents;

*Description*

An item within a container object was added to the selection. The system sends this event for the following user interface elements: list box, list view control, and tree view control. Server applications send this event for their accessible objects. This event signals that a child has been added to an existing selection.

[C#]      public      const      AccessibleEvents      SelectionRemove;

[C++]      public:      const      AccessibleEvents      SelectionRemove;

[VB]      Public      Const      SelectionRemove      As      AccessibleEvents

[JScript]      public      var      SelectionRemove      :      AccessibleEvents;

*Description*

An item within a container object was removed from the selection. The system sends this event for the following user interface elements: list box, list view control, and tree view control. Server applications send this event for their accessible objects. This event signals that a child has been removed from an existing selection.

[C#]      public      const      AccessibleEvents      SelectionWithin;

[C++]      public:      const      AccessibleEvents      SelectionWithin;

[VB]      Public      Const      SelectionWithin      As      AccessibleEvents

[JScript]      public      var      SelectionWithin      :      AccessibleEvents;

*Description*

Numerous selection changes occurred within a container object. The system sends this event for list boxes. Server applications send this event for their accessible objects. This event can be sent when the selected items within a

control have changed substantially. This event informs the client that many selection changes have occurred. This preferable to sending several **SelectionAdd** or **SelectionRemove** events.

[C#]	public	const	AccessibleEvents	Show;
[C++]	public:	const	AccessibleEvents	Show;
[VB]	Public	Const	Show As	AccessibleEvents
[JScript]	public	var	Show :	AccessibleEvents;

#### *Description*

A hidden object is being shown. The system sends this event for the following user interface elements: caret, cursor, and window object. Server applications send this event for their accessible objects. Clients can assume that when this event is sent by a parent object, all child objects have already been displayed. Therefore, server applications do not need to send this event for the child objects.

[C#]	public	const	AccessibleEvents	StateChange;
[C++]	public:	const	AccessibleEvents	StateChange;
[VB]	Public	Const	StateChange As	AccessibleEvents
[JScript]	public	var	StateChange :	AccessibleEvents;

#### *Description*

An object's state has changed. The system sends this event for the following user interface elements: check box, combo box, header control, push button, radio button, scroll bar, toolbar control, tree view control, up-down control, and window object. Server applications send this event for their accessible objects. For example, a state change can occur when a button object has been pressed or released, or when an object is being enabled or disabled. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.



```

1
2 [C#]      public      const      AccessibleEvents      SystemAlert;
3 [C++]     public:     const      AccessibleEvents      SystemAlert;
4 [VB]      Public      Const      SystemAlert      As      AccessibleEvents
5 [JScript] public      var      SystemAlert      :      AccessibleEvents;
6

```

#### *Description*

An alert was generated. Server applications send this event whenever an important user interface change has occurred that a user might need to know about. This event is not sent consistently by the system for dialog box objects. This is a known problem and is being addressed by the accessibility team.

```

11 [C#]      public      const      AccessibleEvents      SystemCaptureEnd;
12 [C++]     public:     const      AccessibleEvents      SystemCaptureEnd;
13 [VB]      Public      Const      SystemCaptureEnd      As      AccessibleEvents
14 [JScript] public      var      SystemCaptureEnd      :      AccessibleEvents;
15

```

#### *Description*

A window has lost mouse capture. This event is sent by the system; servers never send this event.

```

20 [C#]      public      const      AccessibleEvents      SystemCaptureStart;
21 [C++]     public:     const      AccessibleEvents      SystemCaptureStart;
22 [VB]      Public      Const      SystemCaptureStart      As      AccessibleEvents
23 [JScript] public      var      SystemCaptureStart      :      AccessibleEvents;
24

```

#### *Description*

A window is being moved or resized. This event is sent by the system; servers never send this event.

[C#] public const AccessibleEvents SystemContextHelpEnd;

[C++] public: const AccessibleEvents SystemContextHelpEnd;

[VB] Public Const SystemContextHelpEnd As AccessibleEvents

[JScript] public var SystemContextHelpEnd : AccessibleEvents;

#### *Description*

A window exited context-sensitive Help mode. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

[C#] public const AccessibleEvents SystemContextHelpStart;

[C++] public: const AccessibleEvents SystemContextHelpStart;

[VB] Public Const SystemContextHelpStart As AccessibleEvents

[JScript] public var SystemContextHelpStart : AccessibleEvents;

#### *Description*

A window entered context-sensitive Help mode. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

[C#] public const AccessibleEvents SystemDialogEnd;

[C++] public: const AccessibleEvents SystemDialogEnd;

[VB] Public Const SystemDialogEnd As AccessibleEvents

[JScript] public var SystemDialogEnd : AccessibleEvents;

### Description

A dialog box was closed. This event is sent by the system for standard dialog boxes. Servers send this event for custom dialog boxes. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

[C#]	public	const	AccessibleEvents	SystemDialogStart;
[C++]	public:	const	AccessibleEvents	SystemDialogStart;
[VB]	Public	Const	SystemDialogStart	As AccessibleEvents
[JScript]	public	var	SystemDialogStart	: AccessibleEvents;

### Description

A dialog box was displayed. This event is sent by the system for standard dialog boxes. Servers send this event for custom dialog boxes. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

[C#]	public	const	AccessibleEvents	SystemDragDropEnd;
[C++]	public:	const	AccessibleEvents	SystemDragDropEnd;
[VB]	Public	Const	SystemDragDropEnd	As AccessibleEvents
[JScript]	public	var	SystemDragDropEnd	: AccessibleEvents;

### Description

An application is about to exit drag-and-drop mode. Applications that support drag-and-drop operations must send this event; the system does not.

[C#]	public	const	AccessibleEvents	SystemDragDropStart;
------	--------	-------	------------------	----------------------

```

1  [C++]      public:      const      AccessibleEvents      SystemDragDropStart;
2  [VB]       Public      Const      SystemDragDropStart      As      AccessibleEvents
3  [JScript]   public      var      SystemDragDropStart      :      AccessibleEvents;

```

#### *Description*

An application is about to enter drag-and-drop mode. Applications that support drag-and-drop operations must send this event; the system does not.

```

8  [C#]       public      const      AccessibleEvents      SystemForeground;
9  [C++]      public:      const      AccessibleEvents      SystemForeground;
10 [VB]       Public      Const      SystemForeground      As      AccessibleEvents
11 [JScript]   public      var      SystemForeground      :      AccessibleEvents;

```

#### *Description*

The foreground window changed. The system sends this event even if the foreground window is changed to another window in the same thread. Server applications never send this event.

```

17 [C#]       public      const      AccessibleEvents      SystemMenuEnd;
18 [C++]      public:      const      AccessibleEvents      SystemMenuEnd;
19 [VB]       Public      Const      SystemMenuEnd      As      AccessibleEvents
20 [JScript]   public      var      SystemMenuEnd      :      AccessibleEvents;

```

#### *Description*

A menu from the menu bar was closed. The system sends this event for standard menus. Servers send this event for custom menus.

```

1
2 [C#]      public      const      AccessibleEvents      SystemMenuPopupEnd;
3 [C++]     public:     const      AccessibleEvents      SystemMenuPopupEnd;
4 [VB]     Public      Const      SystemMenuPopupEnd      As      AccessibleEvents
5 [JScript] public      var      SystemMenuPopupEnd      :      AccessibleEvents;
6

```

#### *Description*

A pop-up menu was closed. The system sends this event for standard menus. Servers send this event for custom menus. When a pop-up menu is closed, the client receives this message followed almost immediately by the **SystemMenuEnd** event. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

```

12 [C#]      public      const      AccessibleEvents      SystemMenuPopupStart;
13 [C++]     public:     const      AccessibleEvents      SystemMenuPopupStart;
14 [VB]     Public      Const      SystemMenuPopupStart      As      AccessibleEvents
15 [JScript] public      var      SystemMenuPopupStart      :      AccessibleEvents;
16

```

#### *Description*

A pop-up menu was displayed. The system sends this event for standard menus. Servers send this event for custom menus. This event is not sent consistently by the system. This is a known problem and is being addressed by the accessibility team.

```

22 [C#]      public      const      AccessibleEvents      SystemMenuStart;
23 [C++]     public:     const      AccessibleEvents      SystemMenuStart;
24 [VB]     Public      Const      SystemMenuStart      As      AccessibleEvents
25 [JScript] public      var      SystemMenuStart      :      AccessibleEvents;

```

### Description

A menu item on the menu bar was selected. The system sends this event for standard. Servers send this event for custom menus . The system may raise more than one **MenuStart** event that may or may not have a corresponding **MenuEnd** event.

[C#]	public	const	AccessibleEvents	SystemMinimizeEnd;
[C++]	public:	const	AccessibleEvents	SystemMinimizeEnd;
[VB]	Public	Const	SystemMinimizeEnd	As AccessibleEvents
[JScript]	public	var	SystemMinimizeEnd	: AccessibleEvents;

### Description

A window object was minimized or maximized. This event is sent by the system; servers never send this event.

[C#]	public	const	AccessibleEvents	SystemMinimizeStart;
[C++]	public:	const	AccessibleEvents	SystemMinimizeStart;
[VB]	Public	Const	SystemMinimizeStart	As AccessibleEvents
[JScript]	public	var	SystemMinimizeStart	: AccessibleEvents;

### Description

A window object is about to be minimized or maximized. This event is sent by the system; servers never send this event.

[C#]	public	const	AccessibleEvents	SystemMoveSizeEnd;
[C++]	public:	const	AccessibleEvents	SystemMoveSizeEnd;

1 [VB] Public Const SystemMoveSizeEnd As AccessibleEvents

2 [JScript] public var SystemMoveSizeEnd : AccessibleEvents;

3  
4 *Description*

5 The movement or resizing of a window is finished. This event is sent by the  
6 system; servers never send this event.

7 [C#] public const AccessibleEvents SystemMoveSizeStart;

8 [C++] public: const AccessibleEvents SystemMoveSizeStart;

9 [VB] Public Const SystemMoveSizeStart As AccessibleEvents

10 [JScript] public var SystemMoveSizeStart : AccessibleEvents;

11  
12 *Description*

13 A window is being moved or resized. This event is sent by the system; servers  
14 never send this event.

15  
16 [C#] public const AccessibleEvents SystemScrollingEnd;

17 [C++] public: const AccessibleEvents SystemScrollingEnd;

18 [VB] Public Const SystemScrollingEnd As AccessibleEvents

19 [JScript] public var SystemScrollingEnd : AccessibleEvents;

20  
21 *Description*

22 Scrolling has ended on a scroll bar. This event is sent by the system for scroll  
23 bars attached to a window and for standard scroll bar controls. Servers send this  
24 event for custom scroll bars

25 [C#] public const AccessibleEvents SystemScrollingStart;

```

1  [C++]      public:      const      AccessibleEvents      SystemScrollingStart;
2  [VB]      Public      Const      SystemScrollingStart      As      AccessibleEvents
3  [JScript]      public      var      SystemScrollingStart      :      AccessibleEvents;

```

#### *Description*

Scrolling has started on a scroll bar. This event is sent by the system for scroll bars attached to a window and for standard scroll bar controls. Servers send this event for custom scroll bars.

```

9  [C#]      public      const      AccessibleEvents      SystemSound;
10 [C++]      public:      const      AccessibleEvents      SystemSound;
11 [VB]      Public      Const      SystemSound      As      AccessibleEvents
12 [JScript]      public      var      SystemSound      :      AccessibleEvents;

```

#### *Description*

A sound was played. The system sends this event when a system sound, such as for menus, is played even if no sound is audible. This might be caused by lack of a sound file or sound card. Servers send this event if a custom user interface element generates a sound.

```

18 [C#]      public      const      AccessibleEvents      SystemSwitchEnd;
19 [C++]      public:      const      AccessibleEvents      SystemSwitchEnd;
20 [VB]      Public      Const      SystemSwitchEnd      As      AccessibleEvents
21 [JScript]      public      var      SystemSwitchEnd      :      AccessibleEvents;

```

#### *Description*

The user released ALT+TAB. The **SwitchEnd** event is sent by the system; servers never send this event. If only one application is running when the user



presses ALT+TAB, the system sends the **SwitchEnd** event without a corresponding **SwitchStart** event.

[C#]	public	const	AccessibleEvents		SystemSwitchStart;
[C++]	public:	const	AccessibleEvents		SystemSwitchStart;
[VB]	Public	Const	SystemSwitchStart	As	AccessibleEvents
[JScript]	public	var	SystemSwitchStart	:	AccessibleEvents;

#### *Description*

The user pressed ALT+TAB, which activates the switch window. If only one application is running when the user presses ALT+TAB, the system raises the **SwitchEnd** event without a corresponding **SwitchStart** event.

[C#]	public	const	AccessibleEvents		ValueChange;
[C++]	public:	const	AccessibleEvents		ValueChange;
[VB]	Public	Const	ValueChange	As	AccessibleEvents
[JScript]	public	var	ValueChange	:	AccessibleEvents;

#### *Description*

An object's **System.Windows.Forms.AccessibleObject.Value** property changed. The system raises the **ValueChange** event for the following user interface elements: edit control, header control, hot key control, progress bar control, scroll bar, slider control, and up-down control. Server applications send this event for their accessible objects.

## 1. Methods:

AccessibleNavigation enumeration (System.Windows.Forms)

### a) ToString

#### Description

Specifies values for navigating between accessible objects.

Accessible navigation directions are either spatial (up, down, left, and right) or logical (first child, last child, next, and previous). Logical directions are used when clients navigate from one user interface element to another within the same container.

### b) ToString

[C#]	public	const	AccessibleNavigation	Down;
[C++]	public:	const	AccessibleNavigation	Down;
[VB]	Public	Const	Down As	AccessibleNavigation
[JScript]	public	var	Down :	AccessibleNavigation;

#### Description

Navigation to a sibling object located below the starting object.

### c) ToString

[C#]	public	const	AccessibleNavigation	FirstChild;
[C++]	public:	const	AccessibleNavigation	FirstChild;
[VB]	Public	Const	FirstChild As	AccessibleNavigation
[JScript]	public	var	FirstChild :	AccessibleNavigation;

*Description*

Navigation to the first child of the object.

*d) ToString*

[C#]        public        const        AccessibleNavigation        LastChild;

[C++]       public:        const        AccessibleNavigation        LastChild;

[VB]        Public        Const        LastChild        As        AccessibleNavigation

[JScript]    public        var        LastChild        :        AccessibleNavigation;

*Description*

Navigation to the last child of the object.

*e) ToString*

[C#]        public        const        AccessibleNavigation        Left;

[C++]       public:        const        AccessibleNavigation        Left;

[VB]        Public        Const        Left        As        AccessibleNavigation

[JScript]    public        var        Left        :        AccessibleNavigation;

*Description*

Navigation to the sibling object located to the left of the starting object.

*f) ToString*

[C#]        public        const        AccessibleNavigation        Next;

[C++]       public:        const        AccessibleNavigation        Next;

1 [VB] Public Const Next As AccessibleNavigation

2 [JScript] public var Next : AccessibleNavigation;

3  
4 *Description*

5 Navigation to the next logical object, typically from a sibling object to the starting  
6 object.

7 *g) ToString*

8 [C#] public const AccessibleNavigation Previous;

9 [C++] public: const AccessibleNavigation Previous;

10 [VB] Public Const Previous As AccessibleNavigation

11 [JScript] public var Previous : AccessibleNavigation;

12  
13 *Description*

14 Navigation to the previous logical object, typically from a sibling object to the  
15 starting object.

16 *h) ToString*

17  
18 [C#] public const AccessibleNavigation Right;

19 [C++] public: const AccessibleNavigation Right;

20 [VB] Public Const Right As AccessibleNavigation

21 [JScript] public var Right : AccessibleNavigation;

22  
23 *Description*

24 Navigation to the sibling object located to the right of the starting object.

**i) ToString**

[C#]	public	const	AccessibleNavigation	Up;
[C++]	public:	const	AccessibleNavigation	Up;
[VB]	Public	Const	Up	As AccessibleNavigation
[JScript]	public	var	Up	: AccessibleNavigation;

*Description*

Navigation to a sibling object located above the starting object.

AccessibleObject class (System.Windows.Forms)

**a) ToString**

*Description*

Provides an implementation for an object that can be inspected by an accessibility application.

Accessibility applications can adjust features of the application to better serve users who have impairments.

**2. Constructors:**

**a) AccessibleObject**

*Example Syntax:*

**b) ToString**

[C#]	public	AccessibleObject();
[C++]	public:	AccessibleObject();
[VB]	Public	Sub New()

[JScript]                    public                    function                    AccessibleObject();

### *Description*

Initializes a new instance of the **System.Windows.Forms.AccessibleObject** class.

### **3.     Properties:**

*a)     Bounds*

*b)     ToString*

[C#]            public            virtual            Rectangle            Bounds            {get;}

[C++]           public:           \_\_property           virtual           Rectangle           get\_Bounds();

[VB]    Overridable    Public    ReadOnly    Property    Bounds    As    Rectangle

[JScript]       public       function       get       Bounds()       :       Rectangle;

### *Description*

Gets the location and size of the accessible object.

The Bounds property retrieves the object's bounding rectangle. If the object has a non-rectangular shape, then this method returns the smallest rectangle that completely encompasses the entire object region. Therefore, for non-rectangular objects such as list view items, the coordinates of the object's bounding rectangle can fail if tested

**System.Windows.Forms.AccessibleObject.HitTest(System.Int32, System.Int32)** because

**System.Windows.Forms.AccessibleObject.HitTest(System.Int32, System.Int32)** determines the object's boundaries on a pixel-by-pixel basis.

c) *DefaultAction*

d) *ToString*

```
[C#]      public      virtual      string      DefaultAction      {get;}
[C++]     public:     __property     virtual     String*      get_DefaultAction();
[VB]      Overridable Public ReadOnly Property DefaultAction As String
[JScript] public      function      get      DefaultAction()      :      String;
```

*Description*

Gets a string that describes the object's default action. Not all objects have a default action.

The string describes the action that is performed on an object, not what the object does as a result. That is, a toolbar button that prints a document has a default action of "Press" rather than "Prints the current document." Do not confuse an object's default action with its value.

e) *Description*

f) *ToString*

```
[C#]      public      virtual      string      Description      {get;}
[C++]     public:     __property     virtual     String*      get_Description();
[VB]      Overridable Public ReadOnly Property Description As String
[JScript] public      function      get      Description()      :      String;
```

*Description*

Gets a string that describes the visual appearance of the specified object. Not all objects have a description.

This property describes the object's visual appearance to the user.

g) *Help*

h) *ToString*

```
[C#]      public      virtual      string      Help      {get;}
[C++]     public:     __property  virtual      String*    get_Help();
[VB]      Overridable Public  ReadOnly Property Help      As String
[JScript] public      function    get      Help()      :      String;
```

#### *Description*

Gets a description of what the object does or how the object is used.

Not all objects need to support this property. Typically, this property contains balloon-style information (as is found in ToolTips) that is used either to describe what the object does or how to use it. For example, the Help property for a toolbar button that shows a printer might be, "Prints the current document." The text for the Help property does not have to be unique within the user interface. Servers do not need to support the Help property if other properties provide sufficient information about the object's purpose and what actions the object might perform. Accessible objects that expose system-provided controls do not support the Help property.

i) *KeyboardShortcut*

j) *ToString*

```
[C#]      public      virtual      string      KeyboardShortcut      {get;}
[C++]     public:     __property  virtual      String*    get_KeyboardShortcut();
[VB]      Overridable Public  ReadOnly Property KeyboardShortcut As String
[JScript] public      function    get      KeyboardShortcut() :      String;
```

#### *Description*

Gets the shortcut key or access key for the accessible object.



An access key, also known as a mnemonic, is an underlined character in the text of a menu, menu item, or the label of a button, or some other control. For example, a user can display a menu by pressing the ALT key while also pressing the indicated underlined key, such as ALT+F to open the File menu. To use the access key of a menu item, the menu containing the item must be active. Controls such as toolbar buttons and menu items often have an associated shortcut key (also known as a keyboard accelerator). A menu item can have both an access key and a shortcut key associated with it. If this property returns a single character, you can assume it is an access key.

*k) Name*

*l) ToString*

[C#] public virtual string Name {get; set;}

[C++] public: \_\_property virtual String\* get\_Name(); public: \_\_property virtual void set\_Name(String\*);

[VB] Overridable Public Property Name As String

[JScript] public function get Name() : String; public function set Name(String);

### *Description*

Gets or sets the object name.

The **System.Windows.Forms.AccessibleObject.Name** property is a string used by clients to identify, find, or announce an object for the user. To access the name of a child object, you must first call

**System.Windows.Forms.AccessibleObject.GetChild(System.Int32)** with the index of the child whose name you are retrieving.

*m) Parent*

*n) ToString*

[C#] public virtual AccessibleObject Parent {get;}

[C++] public: \_\_property virtual AccessibleObject\* get\_Parent();

1 [VB] Overridable Public ReadOnly Property Parent As AccessibleObject  
2 [JScript] public function get Parent() : AccessibleObject;

4 *Description*

5 Gets the parent of an accessible object.

6 All objects should support this property.

7 *o) Role*

8 *p) ToString*

10 [C#] public virtual AccessibleRole Role {get;}

11 [C++] public: \_\_property virtual AccessibleRole get\_Role();

12 [VB] Overridable Public ReadOnly Property Role As AccessibleRole

13 [JScript] public function get Role() : AccessibleRole;

15 *Description*

16 Gets the role of this accessible object.

17 The role of the object helps describe the function of the object.

18 *q) State*

19 *r) ToString*

21 [C#] public virtual AccessibleStates State {get;}

22 [C++] public: \_\_property virtual AccessibleStates get\_State();

23 [VB] Overridable Public ReadOnly Property State As AccessibleStates

24 [JScript] public function get State() : AccessibleStates;

## Description

Gets the state of this accessible object.

All objects should support this property.

s) *Value*

t) *ToString*

[C#] public virtual string Value {get; set;}

[C++] public: \_\_property virtual String\* get\_Value();public: \_\_property virtual  
void set\_Value(String\*);

[VB] Overridable Public Property Value As String

[JScript] public function get Value() : String;public function set Value(String);

## Description

Gets or sets the value of an accessible object.

Typically, the **System.Windows.Forms.AccessibleObject.Value** property represents visual information contained by the object. Not all objects support the **System.Windows.Forms.AccessibleObject.Value** property.

u) *IAccessible.accDoDefaultAction*

[C#] void IAccessible.accDoDefaultAction(object childID);

[C++] void IAccessible::accDoDefaultAction(Object\* childID);

[VB] Sub accDoDefaultAction(ByVal childID As Object) Implements  
IAccessible.accDoDefaultAction

[JScript] function IAccessible.accDoDefaultAction(childID : Object);

v) *IAccessible.accHitTest*

```

1
2 [C#] object IAccessible.accHitTest(int xLeft, int yTop);
3
4 [C++] Object* IAccessible::accHitTest(int xLeft, int yTop);
5
6 [VB] Function accHitTest(ByVal xLeft As Integer, ByVal yTop As Integer) As
7 Object Implements IAccessible.accHitTest
8
9 [JScript] function IAccessible.accHitTest(xLeft : int, yTop : int) : Object;

```

w) *IAccessible.accLocation*

```

10 [C#] void IAccessible.accLocation(out int pxLeft, out int pyTop, out int pcxWidth,
11 out int pcyHeight, object childID);
12
13 [C++] void IAccessible::accLocation(int* pxLeft, int* pyTop, int* pcxWidth, int*
14 pcyHeight, Object* childID);
15
16 [VB] Sub accLocation(ByRef pxLeft As Integer, ByRef pyTop As Integer, ByRef
17 pcxWidth As Integer, ByRef pcyHeight As Integer, ByVal childID As Object)
18 Implements IAccessible.accLocation
19
20 [JScript] function IAccessible.accLocation(pxLeft : int, pyTop : int, pcxWidth :
21 int, pcyHeight : int, childID : Object);

```

x) *IAccessible.accNavigate*

```

22 [C#] object IAccessible.accNavigate(int navDir, object childID);
23
24 [C++] Object* IAccessible::accNavigate(int navDir, Object* childID);
25
26 [VB] Function accNavigate(ByVal navDir As Integer, ByVal childID As Object)
27 As Object Implements IAccessible.accNavigate
28
29 [JScript] function IAccessible.accNavigate(navDir : int, childID : Object) : Object;

```

y) *IAccessible.accSelect*

```
[C#]    void    IAccessible.accSelect(int    flagsSelect,    object    childID);
[C++]    void    IAccessible::accSelect(int    flagsSelect,    Object*    childID);
[VB] Sub accSelect(ByVal flagsSelect As Integer, ByVal childID As Object)
Implements                                     IAccessible.accSelect
[JScript] function IAccessible.accSelect(flagsSelect : int, childID : Object);
```

z) *IAccessible.get\_accChild*

```
[C#]        object        IAccessible.get_accChild(object        childID);
[C++]        Object*        IAccessible::get_accChild(Object*        childID);
[VB] Function get_accChild(ByVal childID As Object) As Object Implements
IAccessible.get_accChild
[JScript] function IAccessible.get_accChild(childID : Object) : Object;
```

aa) *IAccessible.get\_accDefaultAction*

```
[C#]        string        IAccessible.get_accDefaultAction(object        childID);
[C++]        String*        IAccessible::get_accDefaultAction(Object*        childID);
[VB] Function get_accDefaultAction(ByVal childID As Object) As String
Implements                                     IAccessible.get_accDefaultAction
[JScript] function IAccessible.get_accDefaultAction(childID : Object) : String;
```

bb) *IAccessible.get\_accDescription*

```
[C#]        string        IAccessible.get_accDescription(object        childID);
[C++]        String*        IAccessible::get_accDescription(Object*        childID);
```

1 [VB] Function get\_accDescription(ByVal childID As Object) As String  
2 Implements IAccessible.get\_accDescription

3 [JScript] function IAccessible.get\_accDescription(childID : Object) : String;

4 **cc) IAccessible.get\_accHelp**

6 [C#] string IAccessible.get\_accHelp(object childID);

7 [C++] String\* IAccessible::get\_accHelp(Object\* childID);

8 [VB] Function get\_accHelp(ByVal childID As Object) As String Implements  
9 IAccessible.get\_accHelp

10 [JScript] function IAccessible.get\_accHelp(childID : Object) : String;

11 **dd) IAccessible.get\_accHelpTopic**

13 [C#] int IAccessible.get\_accHelpTopic(out string pszHelpFile, object childID);

14 [C++] int IAccessible::get\_accHelpTopic(String\*\* pszHelpFile, Object\* childID);

15 [VB] Function get\_accHelpTopic(ByRef pszHelpFile As String, ByVal childID  
16 As Object) As Integer Implements IAccessible.get\_accHelpTopic

17 [JScript] function IAccessible.get\_accHelpTopic(pszHelpFile : String, childID :  
18 Object) : int;

19 **ee) IAccessible.get\_accKeyboardShortcut**

21 [C#] string IAccessible.get\_accKeyboardShortcut(object childID);

22 [C++] String\* IAccessible::get\_accKeyboardShortcut(Object\* childID);

23 [VB] Function get\_accKeyboardShortcut(ByVal childID As Object) As String  
24 Implements IAccessible.get\_accKeyboardShortcut

1 [JScript] function IAccessible.get\_accKeyboardShortcut(childID : Object) :  
2 String;

3 *ff) IAccessible.get\_accName*

5 [C#] string IAccessible.get\_accName(object childID);

6 [C++] String\* IAccessible::get\_accName(Object\* childID);

7 [VB] Function get\_accName(ByVal childID As Object) As String Implements

8 IAccessible.get\_accName

9 [JScript] function IAccessible.get\_accName(childID : Object) : String;

10 *gg) IAccessible.get\_accRole*

12 [C#] object IAccessible.get\_accRole(object childID);

13 [C++] Object\* IAccessible::get\_accRole(Object\* childID);

14 [VB] Function get\_accRole(ByVal childID As Object) As Object Implements

15 IAccessible.get\_accRole

16 [JScript] function IAccessible.get\_accRole(childID : Object) : Object;

17 *hh) IAccessible.get\_accState*

19 [C#] object IAccessible.get\_accState(object childID);

20 [C++] Object\* IAccessible::get\_accState(Object\* childID);

21 [VB] Function get\_accState(ByVal childID As Object) As Object Implements

22 IAccessible.get\_accState

23 [JScript] function IAccessible.get\_accState(childID : Object) : Object;

ii) *IAccessible.get\_accValue*

```
[C#]      string      IAccessible.get_accValue(object      childID);  
[C++]     String*     IAccessible::get_accValue(Object*     childID);  
[VB] Function get_accValue(ByVal childID As Object) As String Implements  
IAccessible.get_accValue  
[JScript] function IAccessible.get_accValue(childID : Object) : String;
```

jj) *IAccessible.set\_accName*

```
[C#] void IAccessible.set_accName(object childID, string newName);  
[C++] void IAccessible::set_accName(Object* childID, String* newName);  
[VB] Sub set_accName(ByVal childID As Object, ByVal newName As String)  
Implements IAccessible.set_accName  
[JScript] function IAccessible.set_accName(childID : Object, newName : String);
```

kk) *IAccessible.set\_accValue*

```
[C#] void IAccessible.set_accValue(object childID, string newValue);  
[C++] void IAccessible::set_accValue(Object* childID, String* newValue);  
[VB] Sub set_accValue(ByVal childID As Object, ByVal newValue As String)  
Implements IAccessible.set_accValue  
[JScript] function IAccessible.set_accValue(childID : Object, newValue : String);
```

ll) *DoDefaultAction*

```
[C#]      public      virtual      void      DoDefaultAction();  
[C++]     public:     virtual      void      DoDefaultAction();
```



[VB]	Overridable	Public	Sub	DoDefaultAction()
[JScript]	public	function		DoDefaultAction();

#### Description

Performs the default action associated with this accessible object.

Clients can retrieve the object's default action by inspecting an objects DeafaultAction property. A client could use Automation (if supported) instead of DoDefaultAction to perform an object's default action. However, DoDefaultAction provides an easy way to perform an object's most commonly used action.

#### mm) GetChild

[C#]	public	virtual	AccessibleObject	GetChild(int	index);
[C++]	public:	virtual	AccessibleObject*	GetChild(int	index);
[VB]	Overridable	Public	Function	GetChild(ByVal	index As Integer) As
				AccessibleObject	
[JScript]	public	function	GetChild(index	: int)	: AccessibleObject;

#### Description

Retrieves the accessible child corresponding to the specified index.

**Return Value:** An **System.Windows.Forms.AccessibleObject** that represents the accessible child corresponding to the specified index.

All accesible objects must support this property. If the method is not overridden, it returns **null** . Override this method when an accessible object wants to provide custom accessible children. If the index is invalid, then this method should return **null** . When you override this method, you must also override **System.Windows.Forms.AccessibleObject.GetChildCount** . The zero-based index of the accessible child.

*nn) GetChildCount*

```
[C#]      public      virtual      int      GetChildCount();
[C++]      public:      virtual      int      GetChildCount();
[VB]      Overridable  Public  Function  GetChildCount()  As  Integer
[JScript]  public      function  GetChildCount()      :      int;
```

*Description*

Retrieves the number of children belonging to an accessible object.

*Return Value:* The number of children belonging to an accessible object.

All objects must support this property. The default implementation returns -1. Override this method when an accessible object wants to provide custom accessible children. When you override this method, you must also override **System.Windows.Forms.AccessibleObject.GetChild(System.Int32)**.

*oo) GetFocused*

```
[C#]      public      virtual      AccessibleObject      GetFocused();
[C++]      public:      virtual      AccessibleObject*      GetFocused();
[VB]      Overridable  Public  Function  GetFocused()  As  AccessibleObject
[JScript]  public      function  GetFocused()      :      AccessibleObject;
```

*Description*

Retrieve the object that has the keyboard focus.

*Return Value:* An **System.Windows.Forms.AccessibleObject** that specifies the currently focused child. This method returns the calling object if the object itself is focused. Returns **null** if no object has focus.

The concept of keyboard focus is related to that of an active window. An active window is the foreground window on which the user is working. The object with the keyboard focus is either the active window or a child object of the active window.

pp) *GetHelpTopic*

```
[C#]    public    virtual    int    GetHelpTopic(out    string    fileName);
[C++]    public:    virtual    int    GetHelpTopic(String**    fileName);
[VB]    Overridable Public Function GetHelpTopic(ByRef fileName As String) As
Integer
[JScript]    public    function    GetHelpTopic(fileName    :    String)    :    int;
```

*Description*

Gets an identifier for a Help topic and the path to the Help file associated with this accessible object.

*Return Value:* An identifier for a Help topic, or -1 if there is no Help topic. On return, the *fileName* parameter will contain the path to the Help file associated with this accessible object.

Pass the identifier to the WinHelp file specified by the *fileName* parameter to identify the desired Help topic. On return, this will contain the path to the Help file associated with this accessible object.

qq) *GetSelected*

```
[C#]    public    virtual    AccessibleObject    GetSelected();
[C++]    public:    virtual    AccessibleObject*    GetSelected();
[VB]    Overridable Public Function GetSelected() As AccessibleObject
[JScript]    public    function    GetSelected()    :    AccessibleObject;
```

*Description*

Retrieves the currently selected child.

*Return Value:* An **System.Windows.Forms.AccessibleObject** that represents the currently selected child. This method returns the calling object if the object itself is selected. Returns **null** if there is no currently selected child and the object itself does not have focus.

All objects that can be selected should support this property.

*rr) HitTest*

```
[C#]    public    virtual    AccessibleObject    HitTest(int    x,    int    y);
[C++]    public:    virtual    AccessibleObject*    HitTest(int    x,    int    y);
[VB]    Overridable Public Function HitTest(ByVal x As Integer, ByVal y As
Integer)                                     As                                     AccessibleObject
[JScript] public function HitTest(x : int, y : int) : AccessibleObject;
```

*Description*

Retrieves the child object at the specified screen coordinates.

*Return Value:* An **System.Windows.Forms.AccessibleObject** that represents the child object at the given screen coordinates. This method returns the calling object if the object itself is at the location specified. Returns **null** if there is no object is at the hit tested location.

For non-rectangular objects such as list view items, the coordinates of the object's bounding rectangle retrieved by

**System.Windows.Forms.AccessibleObject.Bounds** can fail if tested with **System.Windows.Forms.AccessibleObject.HitTest(System.Int32, System.Int32)** because

**System.Windows.Forms.AccessibleObject.HitTest(System.Int32, System.Int32)** determines the object's boundaries on a pixel-by-pixel basis. The horizontal screen coordinate. The vertical screen coordinate.

*ss) Navigate*

```
[C#]    public    virtual    AccessibleObject    Navigate(AccessibleNavigation navdir);
[C++]    public:    virtual    AccessibleObject*    Navigate(AccessibleNavigation navdir);
[VB]    Overridable Public Function Navigate(ByVal navdir As
AccessibleNavigation)                       As                       AccessibleObject
[JScript] public function Navigate(navdir : AccessibleNavigation) :
```

AccessibleObject;

### Description

Navigates to another accessible object.

*Return Value:* An **System.Windows.Forms.AccessibleObject** that represents one of the **System.Windows.Forms.AccessibleNavigation** values.

Navigation (both spatial and logical) is always restricted to the user interface elements within a container. With spatial navigation, clients can navigate only to a sibling of the starting object. Depending on the navigational flag used with logical navigation, clients can navigate to either a child or to a sibling of the starting object. This method does not change the selection or focus. To change the focus or to select an object, use

**System.Windows.Forms.AccessibleObject.Select(System.Windows.Forms.AccessibleSelection)** . The **System.Windows.Forms.AccessibleObject.Navigate(System.Windows.Forms.AccessibleNavigation)** method retrieves only user interface elements that have a defined screen location. One of the **System.Windows.Forms.AccessibleNavigation** values.

#### *tt)      Select*

[C#]      public      virtual      void      Select(AccessibleSelection      flags);

[C++]      public:      virtual      void      Select(AccessibleSelection      flags);

[VB]      Overridable      Public      Sub      Select(ByVal flags As AccessibleSelection)

[JScript]      public      function      Select(flags      :      AccessibleSelection);

### Description

Modifies the selection or moves the keyboard focus of the accessible object.

Applications can use this method to perform complex selection operations. One of the **System.Windows.Forms.AccessibleSelection** values.

**uu) IReflect.GetField**

[C#] FieldInfo IReflect.GetField(string name, BindingFlags bindingAttr);  
[C++] FieldInfo\* IReflect::GetField(String\* name, BindingFlags bindingAttr);  
[VB] Function GetField(ByVal name As String, ByVal bindingAttr As  
BindingFlags) As FieldInfo Implements IReflect.GetField  
[JScript] function IReflect.GetField(name : String, bindingAttr : BindingFlags) :  
FieldInfo;

**vv) IReflect.GetFields**

[C#] FieldInfo[] IReflect.GetFields(BindingFlags bindingAttr);  
[C++] FieldInfo\* IReflect::GetFields(BindingFlags bindingAttr) [];  
[VB] Function GetFields(ByVal bindingAttr As BindingFlags) As FieldInfo()  
Implements IReflect.GetFields  
[JScript] function IReflect.GetFields(bindingAttr : BindingFlags) : FieldInfo[];

**ww) IReflect.GetMember**

[C#] MemberInfo[] IReflect.GetMember(string name, BindingFlags bindingAttr);  
[C++] MemberInfo\* IReflect::GetMember(String\* name, BindingFlags  
bindingAttr) [];  
[VB] Function GetMember(ByVal name As String, ByVal bindingAttr As  
BindingFlags) As MemberInfo() Implements IReflect.GetMember  
[JScript] function IReflect.GetMember(name : String, bindingAttr : BindingFlags)  
: MemberInfo[];

xx) *IReflect.GetMembers*

```
1
2 [C#] MemberInfo[] IReflect.GetMembers(BindingFlags bindingAttr);
3 [C++] MemberInfo* IReflect::GetMembers(BindingFlags bindingAttr) [];
4 [VB] Function GetMembers(ByVal bindingAttr As BindingFlags) As
5 MemberInfo() Implements IReflect.GetMembers
6 [JScript] function IReflect.GetMembers(bindingAttr : BindingFlags) :
7 MemberInfo[];
```

yy) *IReflect.GetMethod*

```
10 [C#] MethodInfo IReflect.GetMethod(string name, BindingFlags bindingAttr);
11 [C++] MethodInfo* IReflect::GetMethod(String* name, BindingFlags
12 bindingAttr);
13 [VB] Function GetMethod(ByVal name As String, ByVal bindingAttr As
14 BindingFlags) As MethodInfo Implements IReflect.GetMethod
15 [JScript] function IReflect.GetMethod(name : String, bindingAttr : BindingFlags) :
16 MethodInfo;
```

zz) *IReflect.GetMethod*

```
18
19
20 [C#] MethodInfo IReflect.GetMethod(string name, BindingFlags bindingAttr,
21 Binder binder, Type[] types, ParameterModifier[] modifiers);
22 [C++] MethodInfo* IReflect::GetMethod(String* name, BindingFlags
23 bindingAttr, Binder* binder, Type* types[], ParameterModifier modifiers[]);
24 [VB] Function GetMethod(ByVal name As String, ByVal bindingAttr As
25 BindingFlags, ByVal binder As Binder, ByVal types() As Type, ByVal
```

1 modifiers() As ParameterModifier) As MethodInfo Implements

2 IReflect.GetMethod

3 [JScript] function IReflect.GetMethod(name : String, bindingAttr : BindingFlags,

4 binder : Binder, types : Type[], modifiers : ParameterModifier[]) : MethodInfo;

5 **aaa) IReflect.GetMethods**

6  
7 [C#] MethodInfo[] IReflect.GetMethods(BindingFlags bindingAttr);

8 [C++] MethodInfo\* IReflect::GetMethods(BindingFlags bindingAttr) [];

9 [VB] Function GetMethods(ByVal bindingAttr As BindingFlags) As

10 MethodInfo() Implements IReflect.GetMethods

11 [JScript] function IReflect.GetMethods(bindingAttr : BindingFlags) :

12 MethodInfo[];

13 **bbb) IReflect.GetProperties**

14  
15 [C#] PropertyInfo[] IReflect.GetProperties(BindingFlags bindingAttr);

16 [C++] PropertyInfo\* IReflect::GetProperties(BindingFlags bindingAttr) [];

17 [VB] Function GetProperties(ByVal bindingAttr As BindingFlags) As

18 PropertyInfo() Implements IReflect.GetProperties

19 [JScript] function IReflect.GetProperties(bindingAttr : BindingFlags) :

20 PropertyInfo[];

21 **ccc) IReflect.GetProperty**

22  
23 [C#] PropertyInfo IReflect.GetProperty(string name, BindingFlags bindingAttr);

24 [C++] PropertyInfo\* IReflect::GetProperty(String\* name, BindingFlags



```

1 bindingAttr);
2 [VB] Function GetProperty(ByVal name As String, ByVal bindingAttr As
3 BindingFlags) As PropertyInfo Implements IReflect.GetProperty
4 [JScript] function IReflect.GetProperty(name : String, bindingAttr : BindingFlags)
5 : PropertyInfo;

```

**ddd) IReflect.GetProperty**

```

8 [C#] PropertyInfo IReflect.GetProperty(string name, BindingFlags bindingAttr,
9 Binder binder, Type returnType, Type[] types, ParameterModifier[] modifiers);
10 [C++] PropertyInfo* IReflect::GetProperty(String* name, BindingFlags
11 bindingAttr, Binder* binder, Type* returnType, Type* types[], ParameterModifier
12 modifiers[]);

```

```

13 [VB] Function GetProperty(ByVal name As String, ByVal bindingAttr As
14 BindingFlags, ByVal binder As Binder, ByVal returnType As Type, ByVal
15 types() As Type, ByVal modifiers() As ParameterModifier) As PropertyInfo
16 Implements IReflect.GetProperty
17 [JScript] function IReflect.GetProperty(name : String, bindingAttr : BindingFlags,
18 binder : Binder, returnType : Type, types : Type[], modifiers :
19 ParameterModifier[]) : PropertyInfo;

```

**eee) IReflect.InvokeMember**

```

22 [C#] object IReflect.InvokeMember(string name, BindingFlags invokeAttr, Binder
23 binder, object target, object[] args, ParameterModifier[] modifiers, CultureInfo
24 culture, string[] namedParameters);

```

```

1 [C++] Object* IReflect::InvokeMember(String* name, BindingFlags invokeAttr,
2 Binder* binder, Object* target, Object* args __gc[], ParameterModifier
3 modifiers[], CultureInfo* culture, String* namedParameters __gc[]);
4 [VB] Function InvokeMember(ByVal name As String, ByVal invokeAttr As
5 BindingFlags, ByVal binder As Binder, ByVal target As Object, ByVal args() As
6 Object, ByVal modifiers() As ParameterModifier, ByVal culture As CultureInfo,
7 ByVal namedParameters() As String) As Object Implements
8 IReflect.InvokeMember
9 [JScript] function IReflect.InvokeMember(name : String, invokeAttr :
10 BindingFlags, binder : Binder, target : Object, args : Object[], modifiers :
11 ParameterModifier[], culture : CultureInfo, namedParameters : String[]) : Object;
12
13
14 [C#] void
15 UnsafeNativeMethods.IEnumVariant.Clone(UnsafeNativeMethods.IEnumVariant
16 [] v);
17 [C++] void
18 UnsafeNativeMethods::IEnumVariant::Clone(UnsafeNativeMethods.IEnumVaria
19 nt* v[]);
20 [VB] Sub IEnumVariant.Clone(ByVal v() As
21 UnsafeNativeMethods.IEnumVariant) Implements
22 UnsafeNativeMethods.IEnumVariant.Clone
23 [JScript] function UnsafeNativeMethods.IEnumVariant.Clone(v :
24 UnsafeNativeMethods.IEnumVariant[]);
25

```

**ggg) UnsafeNativeMethods.IEnumVariant.Next**

[C#] int UnsafeNativeMethods.IEnumVariant.Next(int n, IntPtr rgvar, int[] ns);

[C++] int UnsafeNativeMethods::IEnumVariant::Next(int n, IntPtr rgvar, int ns  
\_\_gc[]);

[VB] Function IEnumVariant.Next(ByVal n As Integer, ByVal rgvar As IntPtr,  
ByVal ns() As Integer) As Integer Implements  
UnsafeNativeMethods.IEnumVariant.Next

[JScript] function UnsafeNativeMethods.IEnumVariant.Next(n : int, rgvar : IntPtr,  
ns : int[]) : int;

**hhh) UnsafeNativeMethods.IEnumVariant.Reset**

[C#] void UnsafeNativeMethods.IEnumVariant.Reset();

[C++] void UnsafeNativeMethods::IEnumVariant::Reset();

[VB] Sub IEnumVariant.Reset() Implements  
UnsafeNativeMethods.IEnumVariant.Reset

[JScript] function UnsafeNativeMethods.IEnumVariant.Reset();

**iii) UnsafeNativeMethods.IEnumVariant.Skip**

[C#] void UnsafeNativeMethods.IEnumVariant.Skip(int n);

[C++] void UnsafeNativeMethods::IEnumVariant::Skip(int n);

[VB] Sub IEnumVariant.Skip(ByVal n As Integer) Implements  
UnsafeNativeMethods.IEnumVariant.Skip

[JScript] function UnsafeNativeMethods.IEnumVariant.Skip(n : int);

### *jjj) UseStdAccessibleObjects*

```
[C#]    protected    void    UseStdAccessibleObjects(IntPtr    handle);
[C++]    protected:    void    UseStdAccessibleObjects(IntPtr    handle);
[VB]    Protected    Sub    UseStdAccessibleObjects(ByVal    handle    As    IntPtr)
[JScript] protected function UseStdAccessibleObjects(handle : IntPtr); Associates
an object with an an instance of an System.Windows.Forms.AccessibleObject .
```

#### *Description*

Associates an object with an an instance of an **System.Windows.Forms.AccessibleObject** based on the handle of the object.

Server applications can call this function when they contain a custom UI object that is similar to a system-provided object. Server applications call `CreateStdAccessibleObject` and override the `IAccessible` methods and properties as needed to match their custom object. This approach saves server developers the work of fully implementing all the `IAccessible` properties and methods. This function is similar to `CreateStdAccessibleProxy`, except that `CreateStdAccessibleProxy` allows you to specify the class name as a parameter whereas `CreateStdAccessibleObject` uses the class name associated with the `hwnd`. An **System.IntPtr** that contains the handle of the object.

### *kkk) UseStdAccessibleObjects*

```
[C#]    protected    void    UseStdAccessibleObjects(IntPtr    handle,    int    objid);
[C++]    protected:    void    UseStdAccessibleObjects(IntPtr    handle,    int    objid);
[VB]    Protected    Sub    UseStdAccessibleObjects(ByVal    handle    As    IntPtr,    ByVal
objid
As
Integer)
[JScript] protected function UseStdAccessibleObjects(handle : IntPtr, objid : int);
```

## Description

Associates an object with an instance of an **System.Windows.Forms.AccessibleObject** based on the handle and the object id of the object.

Server applications can call this function when they contain a custom UI object that is similar to a system-provided object. Server applications call `CreateStdAccessibleObject` and override the `IAccessible` methods and properties as needed to match their custom object. This approach saves server developers the work of fully implementing all the `IAccessible` properties and methods. An **System.IntPtr** that contains the handle of the object. An `Int` that defines the type of object that the *handle* parameter refers to.

AccessibleRole enumeration (System.Windows.Forms)

### a) *UseStdAccessibleObjects*

## Description

Specifies values representing possible roles for an accessible object.

The role of the object is used to determine the actions taken by the accessible application to make the accessible.

### b) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Alert;
[C++]	public:	const	AccessibleRole	Alert;
[VB]	Public	Const	Alert	As AccessibleRole
[JScript]	public	var	Alert	: AccessibleRole;

## Description

An alert or condition that a user should be notified about. This role should be used only for objects that embody an alert but are not associated with another user interface element, such as a message box, graphic, text, or sound.

*c) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Animation;
[C++]	public:	const	AccessibleRole	Animation;
[VB]	Public	Const	Animation	As AccessibleRole
[JScript]	public	var	Animation	: AccessibleRole;

*Description*

An animation control, which contains content that is changing over time, such as a control that displays a series of bitmap frames, like a film strip. Animation controls are usually displayed when files are being copied, or when some other time-consuming task is being performed.

*d) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Application;
[C++]	public:	const	AccessibleRole	Application;
[VB]	Public	Const	Application	As AccessibleRole
[JScript]	public	var	Application	: AccessibleRole;

*Description*

The main window for an application.

*e) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Border;
------	--------	-------	----------------	---------

```

1  [C++]      public:      const      AccessibleRole      Border;
2  [VB]      Public      Const      Border      As      AccessibleRole
3  [JScript]      public      var      Border      :      AccessibleRole;

```

#### *Description*

A window border. The entire border is represented by a single object, rather than by separate objects for each side.

#### *f) UseStdAccessibleObjects*

```

9  [C#]      public      const      AccessibleRole      ButtonDropDown;
10 [C++]      public:      const      AccessibleRole      ButtonDropDown;
11 [VB]      Public      Const      ButtonDropDown      As      AccessibleRole
12 [JScript]      public      var      ButtonDropDown      :      AccessibleRole;

```

#### *Description*

A button that drops down a list of items.

#### *g) UseStdAccessibleObjects*

```

18 [C#]      public      const      AccessibleRole      ButtonDropDownGrid;
19 [C++]      public:      const      AccessibleRole      ButtonDropDownGrid;
20 [VB]      Public      Const      ButtonDropDownGrid      As      AccessibleRole
21 [JScript]      public      var      ButtonDropDownGrid      :      AccessibleRole;

```

#### *Description*

A button that drops down a grid.

**h) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	ButtonMenu;
[C++]	public:	const	AccessibleRole	ButtonMenu;
[VB]	Public	Const	ButtonMenu	As AccessibleRole
[JScript]	public	var	ButtonMenu	: AccessibleRole;

*Description*

A button that drops down a menu.

**i) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Caret;
[C++]	public:	const	AccessibleRole	Caret;
[VB]	Public	Const	Caret	As AccessibleRole
[JScript]	public	var	Caret	: AccessibleRole;

*Description*

A caret, which is a flashing line, block, or bitmap that marks the location of the insertion point in a window's client area.

**j) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Cell;
[C++]	public:	const	AccessibleRole	Cell;
[VB]	Public	Const	Cell	As AccessibleRole
[JScript]	public	var	Cell	: AccessibleRole;



*Description*

A cell within a table.

**k) UseStdAccessibleObjects**

[C#]            public            const            AccessibleRole            Character;

[C++]           public:            const            AccessibleRole            Character;

[VB]            Public            Const            Character            As            AccessibleRole

[JScript]       public            var            Character            :            AccessibleRole;

*Description*

A cartoon-like graphic object, such as Microsoft Office Assistant, which is typically displayed to provide help to users of an application.

**l) UseStdAccessibleObjects**

[C#]            public            const            AccessibleRole            Chart;

[C++]           public:            const            AccessibleRole            Chart;

[VB]            Public            Const            Chart            As            AccessibleRole

[JScript]       public            var            Chart            :            AccessibleRole;

*Description*

A graphical image used to represent data.

**m) UseStdAccessibleObjects**

[C#]            public            const            AccessibleRole            CheckButton;

[C++]	public:	const	AccessibleRole	CheckBox;
[VB]	Public	Const	CheckBox	As AccessibleRole
[JScript]	public	var	CheckBox	: AccessibleRole;

*Description*

A check box control, which is an option that can be turned on or off independently of other options.

*n) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Client;
[C++]	public:	const	AccessibleRole	Client;
[VB]	Public	Const	Client	As AccessibleRole
[JScript]	public	var	Client	: AccessibleRole;

*Description*

A window's user area.

*o) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Clock;
[C++]	public:	const	AccessibleRole	Clock;
[VB]	Public	Const	Clock	As AccessibleRole
[JScript]	public	var	Clock	: AccessibleRole;

*Description*

A control that displays the time.

**p) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Column;
[C++]	public:	const	AccessibleRole	Column;
[VB]	Public	Const	Column	As AccessibleRole
[JScript]	public	var	Column	: AccessibleRole;

**Description**

A column of cells within a table.

**q) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	ColumnHeader;
[C++]	public:	const	AccessibleRole	ColumnHeader;
[VB]	Public	Const	ColumnHeader	As AccessibleRole
[JScript]	public	var	ColumnHeader	: AccessibleRole;

**Description**

A column header, which provides a visual label for a column in a table.

**r) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	ComboBox;
[C++]	public:	const	AccessibleRole	ComboBox;
[VB]	Public	Const	ComboBox	As AccessibleRole
[JScript]	public	var	ComboBox	: AccessibleRole;

*Description*

A combo box, which is an edit control with an associated list box that provides a set of predefined choices.

*s) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Cursor;
[C++]	public:	const	AccessibleRole	Cursor;
[VB]	Public	Const	Cursor	As AccessibleRole
[JScript]	public	var	Cursor	: AccessibleRole;

*Description*

A mouse pointer.

*t) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Default;
[C++]	public:	const	AccessibleRole	Default;
[VB]	Public	Const	Default	As AccessibleRole
[JScript]	public	var	Default	: AccessibleRole;

*Description*

A system provided role.

*u) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Diagram;
------	--------	-------	----------------	----------

[C++]	public:	const	AccessibleRole	Diagram;
[VB]	Public	Const	Diagram	As AccessibleRole
[JScript]	public	var	Diagram	: AccessibleRole;

*Description*

A graphical image used to diagram data.

*v) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Dial;
[C++]	public:	const	AccessibleRole	Dial;
[VB]	Public	Const	Dial	As AccessibleRole
[JScript]	public	var	Dial	: AccessibleRole;

*Description*

A dial or knob. This can also be a read-only object, like a speedometer.

*w) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Dialog;
[C++]	public:	const	AccessibleRole	Dialog;
[VB]	Public	Const	Dialog	As AccessibleRole
[JScript]	public	var	Dialog	: AccessibleRole;

*Description*

A dialog box or message box.

x) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Document;
[C++]	public:	const	AccessibleRole	Document;
[VB]	Public	Const	Document	As AccessibleRole
[JScript]	public	var	Document	: AccessibleRole;

*Description*

A document window, which is always contained within an application window. This role applies only to multiple document interface (MDI) windows and refers to an object that contains the MDI title bar.

y) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	DropList;
[C++]	public:	const	AccessibleRole	DropList;
[VB]	Public	Const	DropList	As AccessibleRole
[JScript]	public	var	DropList	: AccessibleRole;

*Description*

A drop-down list box. This control shows one item and allows the user to display and select another from a list of alternative choices.

z) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Equation;
[C++]	public:	const	AccessibleRole	Equation;
[VB]	Public	Const	Equation	As AccessibleRole

[JScript]      public      var      Equation      :      AccessibleRole;

*Description*

A mathematical equation.

**aa)      UseStdAccessibleObjects**

[C#]      public      const      AccessibleRole      Graphic;

[C++]      public:      const      AccessibleRole      Graphic;

[VB]      Public      Const      Graphic      As      AccessibleRole

[JScript]      public      var      Graphic      :      AccessibleRole;

*Description*

A picture.

**bb)      UseStdAccessibleObjects**

[C#]      public      const      AccessibleRole      Grip;

[C++]      public:      const      AccessibleRole      Grip;

[VB]      Public      Const      Grip      As      AccessibleRole

[JScript]      public      var      Grip      :      AccessibleRole;

*Description*

A special mouse pointer, which allows a user to manipulate user interface elements such as a window. For example, a user can click and drag a sizing grip in the lower-right corner of a window to resize it.

**cc) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Grouping;
[C++]	public:	const	AccessibleRole	Grouping;
[VB]	Public	Const	Grouping As	AccessibleRole
[JScript]	public	var	Grouping :	AccessibleRole;

**Description**

Objects grouped in a logical manner. There can be a parent-child relationship between the grouping object and the objects it contains.

**dd) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	HelpBalloon;
[C++]	public:	const	AccessibleRole	HelpBalloon;
[VB]	Public	Const	HelpBalloon As	AccessibleRole
[JScript]	public	var	HelpBalloon :	AccessibleRole;

**Description**

A Help display in the form of a ToolTip or Help balloon, which contains buttons and labels that users can click to open custom Help topics.

**ee) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	HotkeyField;
[C++]	public:	const	AccessibleRole	HotkeyField;
[VB]	Public	Const	HotkeyField As	AccessibleRole
[JScript]	public	var	HotkeyField :	AccessibleRole;



## Description

A hot-key field that allows the user to enter a combination or sequence of keystrokes to be used as a hot key, which enables users to perform an action quickly. A hot-key control displays the keystrokes entered by the user and ensures that the user selects a valid key combination.

### *ff) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Indicator;
[C++]	public:	const	AccessibleRole	Indicator;
[VB]	Public	Const	Indicator	As AccessibleRole
[JScript]	public	var	Indicator	: AccessibleRole;

## Description

An indicator, such as a pointer graphic, that points to the current item.

### *gg) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Link;
[C++]	public:	const	AccessibleRole	Link;
[VB]	Public	Const	Link	As AccessibleRole
[JScript]	public	var	Link	: AccessibleRole;

## Description

A link, which is a connection between a source document and a destination document. This object might look like text or a graphic, but it acts like a button.

**hh) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	List;
[C++]	public:	const	AccessibleRole	List;
[VB]	Public	Const	List As	AccessibleRole
[JScript]	public	var	List :	AccessibleRole;

**Description**

A list box, which allows the user to select one or more items.

**ii) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	ListItem;
[C++]	public:	const	AccessibleRole	ListItem;
[VB]	Public	Const	ListItem As	AccessibleRole
[JScript]	public	var	ListItem :	AccessibleRole;

**Description**

An item in a list box or the list portion of a combo box, drop-down list box, or drop-down combo box.

**jj) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	MenuBar;
[C++]	public:	const	AccessibleRole	MenuBar;
[VB]	Public	Const	MenuBar As	AccessibleRole
[JScript]	public	var	MenuBar :	AccessibleRole;

## Description

A menu bar, usually beneath the title bar of a window, from which menus can be selected by the user.

### kk) UseStdAccessibleObjects

[C#]	public	const	AccessibleRole	MenuItem;
[C++]	public:	const	AccessibleRole	MenuItem;
[VB]	Public	Const	MenuItem	As AccessibleRole
[JScript]	public	var	MenuItem	: AccessibleRole;

## Description

A menu item, which is an entry in a menu that a user can choose to carry out a command, select an option, or display another menu. Functionally, a menu item can be equivalent to a push button, radio button, check box, or menu.

### ll) UseStdAccessibleObjects

[C#]	public	const	AccessibleRole	MenuPopup;
[C++]	public:	const	AccessibleRole	MenuPopup;
[VB]	Public	Const	MenuPopup	As AccessibleRole
[JScript]	public	var	MenuPopup	: AccessibleRole;

## Description

A menu, which presents a list of options from which the user can make a selection to perform an action. All menu types must have this role, including drop-down menus that are displayed by selection from a menu bar, and shortcut menus that are displayed when the right mouse button is clicked.

*mm) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	None;
[C++]	public:	const	AccessibleRole	None;
[VB]	Public	Const	None	As AccessibleRole
[JScript]	public	var	None	: AccessibleRole;

*Description*

No role.

*nn) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Outline;
[C++]	public:	const	AccessibleRole	Outline;
[VB]	Public	Const	Outline	As AccessibleRole
[JScript]	public	var	Outline	: AccessibleRole;

*Description*

An outline or tree structure, such as a tree view control, which displays a hierarchical list and usually allows the user to expand and collapse branches.

*oo) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	OutlineItem;
[C++]	public:	const	AccessibleRole	OutlineItem;
[VB]	Public	Const	OutlineItem	As AccessibleRole
[JScript]	public	var	OutlineItem	: AccessibleRole;

*Description*

An item in an outline or tree structure.

*pp) UseStdAccessibleObjects*

[C#]            public            const            AccessibleRole            PageTab;

[C++]           public:            const            AccessibleRole            PageTab;

[VB]            Public            Const            PageTab            As            AccessibleRole

[JScript]       public            var            PageTab            :            AccessibleRole;

*Description*

A property page that allows a user to view the attributes for a page, such as the page's title, whether it is a home page, or whether the page has been modified. Normally the only child of this control is a grouped object that contains the contents of the associated page.

*qq) UseStdAccessibleObjects*

[C#]            public            const            AccessibleRole            PageTabList;

[C++]           public:            const            AccessibleRole            PageTabList;

[VB]            Public            Const            PageTabList            As            AccessibleRole

[JScript]       public            var            PageTabList            :            AccessibleRole;

*Description*

A container of page tab controls.

*rr) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Pane;
[C++]	public:	const	AccessibleRole	Pane;
[VB]	Public	Const	Pane	As AccessibleRole
[JScript]	public	var	Pane	: AccessibleRole;

*Description*

One of the separate areas in a frame, a split document window, or a rectangular area of the status bar that can be used to display information. Users can navigate between panes and within the contents of the current pane, but cannot navigate between items in different panes. Thus, panes represent a level of grouping lower than frame windows or documents, but above individual controls. Typically the user navigates between panes by pressing TAB, F6, or CTRL+TAB, depending on the context.

*ss) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	ProgressBar;
[C++]	public:	const	AccessibleRole	ProgressBar;
[VB]	Public	Const	ProgressBar	As AccessibleRole
[JScript]	public	var	ProgressBar	: AccessibleRole;

*Description*

A progress bar, which indicates the progress of a lengthy operation by displaying a colored bar inside a horizontal rectangle. The length of the bar in relation to the length of the rectangle corresponds to the percentage of the operation that is complete. This control does not take user input.

tt) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	PropertyPage;
[C++]	public:	const	AccessibleRole	PropertyPage;
[VB]	Public	Const	PropertyPage	As AccessibleRole
[JScript]	public	var	PropertyPage	: AccessibleRole;

*Description*

A property page, which is a dialog box that controls the appearance and the behavior of an object, such as a file or resource. A property page's appearance differs according to its purpose.

uu) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	PushButton;
[C++]	public:	const	AccessibleRole	PushButton;
[VB]	Public	Const	PushButton	As AccessibleRole
[JScript]	public	var	PushButton	: AccessibleRole;

*Description*

A push button control, which is a small rectangular control that a user can turn on or off. A push button, also known as a command button, has a raised appearance in its default off state and a sunken appearance when it is turned on.

vv) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	RadioButton;
[C++]	public:	const	AccessibleRole	RadioButton;

1 [VB] Public Const RadioButton As AccessibleRole

2 [JScript] public var RadioButton : AccessibleRole;

4 *Description*

5 An option button, also known as a radio button. All objects sharing a single  
6 parent that have this attribute are assumed to be part of a single mutually  
7 exclusive group. You can use grouped objects to divide option buttons into  
8 separate groups when necessary.

9 *ww) UseStdAccessibleObjects*

10 [C#] public const AccessibleRole Row;

11 [C++] public: const AccessibleRole Row;

12 [VB] Public Const Row As AccessibleRole

13 [JScript] public var Row : AccessibleRole;

14 *Description*

15 A row of cells within a table.

16 *xx) UseStdAccessibleObjects*

17 [C#] public const AccessibleRole RowHeader;

18 [C++] public: const AccessibleRole RowHeader;

19 [VB] Public Const RowHeader As AccessibleRole

20 [JScript] public var RowHeader : AccessibleRole;

21 *Description*

22 A row header, which provides a visual label for a table row.



**yy) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	ScrollBar;
[C++]	public:	const	AccessibleRole	ScrollBar;
[VB]	Public	Const	ScrollBar	As AccessibleRole
[JScript]	public	var	ScrollBar	: AccessibleRole;

**Description**

A vertical or horizontal scroll bar, which can be either part of the client area or used in a control.

**zz) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Separator;
[C++]	public:	const	AccessibleRole	Separator;
[VB]	Public	Const	Separator	As AccessibleRole
[JScript]	public	var	Separator	: AccessibleRole;

**Description**

Visually divides a space into two regions, such as a separator menu item or a bar dividing split panes within a window.

**aaa) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Slider;
[C++]	public:	const	AccessibleRole	Slider;
[VB]	Public	Const	Slider	As AccessibleRole
[JScript]	public	var	Slider	: AccessibleRole;

## Description

A control, sometimes called a trackbar, that allows a user to adjust a setting in given increments between minimum and maximum values by moving a slider. The volume controls in the Windows operating system are slider controls.

### *bbb) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	Sound;
[C++]	public:	const	AccessibleRole	Sound;
[VB]	Public	Const	Sound	As AccessibleRole
[JScript]	public	var	Sound	: AccessibleRole;

## Description

A system sound, which is associated with various system events.

### *ccc) UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	SpinButton;
[C++]	public:	const	AccessibleRole	SpinButton;
[VB]	Public	Const	SpinButton	As AccessibleRole
[JScript]	public	var	SpinButton	: AccessibleRole;

## Description

A spin box, also known as an up-down control, which contains a pair of arrow buttons that a user click with a mouse to increment or decrement a value. A spin button control is most often used with a companion control, called a buddy window, where the current value is displayed.

**ddd) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	StaticText;
[C++]	public:	const	AccessibleRole	StaticText;
[VB]	Public	Const	StaticText	As AccessibleRole
[JScript]	public	var	StaticText	: AccessibleRole;

**Description**

Read-only text, such as in a label, for other controls or instructions in a dialog box. Static text cannot be modified or selected.

**eee) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	StatusBar;
[C++]	public:	const	AccessibleRole	StatusBar;
[VB]	Public	Const	StatusBar	As AccessibleRole
[JScript]	public	var	StatusBar	: AccessibleRole;

**Description**

A status bar, which is an area typically at the bottom of an application window that displays information about the current operation, state of the application, or selected object. The status bar can have multiple fields that display different kinds of information, such as an explanation of the currently selected menu command in the status bar.

**fff) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Table;
[C++]	public:	const	AccessibleRole	Table;

[VB]	Public	Const	Table	As	AccessibleRole
[JScript]	public	var	Table	:	AccessibleRole;

#### Description

A table containing rows and columns of cells, and optionally, row headers and column headers.

**ggg) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	Text;	
[C++]	public:	const	AccessibleRole	Text;	
[VB]	Public	Const	Text	As	AccessibleRole
[JScript]	public	var	Text	:	AccessibleRole;

#### Description

Selectable text that can be editable or read-only.

**hhh) UseStdAccessibleObjects**

[C#]	public	const	AccessibleRole	TitleBar;	
[C++]	public:	const	AccessibleRole	TitleBar;	
[VB]	Public	Const	TitleBar	As	AccessibleRole
[JScript]	public	var	TitleBar	:	AccessibleRole;

#### Description

A title or caption bar for a window.

iii) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	ToolBar;
[C++]	public:	const	AccessibleRole	ToolBar;
[VB]	Public	Const	ToolBar	As AccessibleRole
[JScript]	public	var	ToolBar	: AccessibleRole;

*Description*

A toolbar, which is a grouping of controls that provide easy access to frequently used features.

jjj) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	ToolTip;
[C++]	public:	const	AccessibleRole	ToolTip;
[VB]	Public	Const	ToolTip	As AccessibleRole
[JScript]	public	var	ToolTip	: AccessibleRole;

*Description*

A tool tip, which is a small rectangular pop-up window that displays a brief description of a command bar button's purpose.

kkk) *UseStdAccessibleObjects*

[C#]	public	const	AccessibleRole	WhiteSpace;
[C++]	public:	const	AccessibleRole	WhiteSpace;
[VB]	Public	Const	WhiteSpace	As AccessibleRole
[JScript]	public	var	WhiteSpace	: AccessibleRole;

*Description*

A blank space between other objects.

**III) UseStdAccessibleObjects**

[C#]            public            const            AccessibleRole            Window;

[C++]           public:            const            AccessibleRole            Window;

[VB]            Public            Const            Window            As            AccessibleRole

[JScript]       public            var            Window            :            AccessibleRole;

*Description*

A window frame, which usually contains child objects such as a title bar, client, and other objects typically contained in a window.

AccessibleSelection enumeration (System.Windows.Forms)

**a) ToString**

*Description*

Specifies how an accessible object is selected or receives focus.

A focused object is the one object that receives keyboard input. The object with the keyboard focus is either the active window or a child object of the active window. A selected object is marked to participate in some type of group operation.

**b) ToString**

[C#]            public            const            AccessibleSelection            AddSelection;

```

1 [C++]      public:      const      AccessibleSelection      AddSelection;
2 [VB]       Public      Const      AddSelection      As      AccessibleSelection
3 [JScript]  public      var      AddSelection      :      AccessibleSelection;

```

#### *Description*

Adds the object to the selection.

#### *c) ToString*

```

9 [C#]       public      const      AccessibleSelection      ExtendSelection;
10 [C++]      public:      const      AccessibleSelection      ExtendSelection;
11 [VB]       Public      Const      ExtendSelection      As      AccessibleSelection
12 [JScript]  public      var      ExtendSelection      :      AccessibleSelection;

```

#### *Description*

Selects all objects between the anchor and the selected object.

#### *d) ToString*

```

18 [C#]       public      const      AccessibleSelection      None;
19 [C++]      public:      const      AccessibleSelection      None;
20 [VB]       Public      Const      None      As      AccessibleSelection
21 [JScript]  public      var      None      :      AccessibleSelection;

```

#### *Description*

The selection or focus of an object is unchanged.

e) *ToString*

```
[C#]      public      const      AccessibleSelection      RemoveSelection;
[C++]     public:     const      AccessibleSelection      RemoveSelection;
[VB]      Public      Const      RemoveSelection      As      AccessibleSelection
[JScript] public      var      RemoveSelection      :      AccessibleSelection;
```

*Description*

Removes the object from the selection.

f) *ToString*

```
[C#]      public      const      AccessibleSelection      TakeFocus;
[C++]     public:     const      AccessibleSelection      TakeFocus;
[VB]      Public      Const      TakeFocus      As      AccessibleSelection
[JScript] public      var      TakeFocus      :      AccessibleSelection;
```

*Description*

Assigns focus to an object and makes it the anchor, which is the starting point for the selection. Can be combined with **TakeSelection** , **ExtendSelection** , **AddSelection** , or **RemoveSelection** .

g) *ToString*

```
[C#]      public      const      AccessibleSelection      TakeSelection;
[C++]     public:     const      AccessibleSelection      TakeSelection;
[VB]      Public      Const      TakeSelection      As      AccessibleSelection
[JScript] public      var      TakeSelection      :      AccessibleSelection;
```



1  
2 *Description*

3 Selects the object and deselects all other objects in the container.

4 AccessibleStates enumeration (System.Windows.Forms)

5 *a) ToString*

6  
7  
8 *Description*

9 Specifies values representing possible states for an accessible object.

10 An accessible object can be associated with one or more of these states.

11 *b) ToString*

12  
13 [C#] public const AccessibleStates AlertHigh;

14 [C++] public: const AccessibleStates AlertHigh;

15 [VB] Public Const AlertHigh As AccessibleStates

16 [JScript] public var AlertHigh : AccessibleStates;

17  
18 *Description*

19 Indicates important information that should be conveyed to the user  
20 immediately. For example, a battery level indicator reaching a critical low level  
21 would transition to this state, in which case a blind access utility would announce  
22 this information immediately to the user, and a screen magnification program  
23 would scroll the screen so that the battery indicator is in view. This state is also  
24 appropriate for any prompt or operation that must be completed before the user  
25 can continue.

c) *ToString*

[C#]	public	const	AccessibleStates	AlertLow;
[C++]	public:	const	AccessibleStates	AlertLow;
[VB]	Public	Const	AlertLow	As AccessibleStates
[JScript]	public	var	AlertLow	: AccessibleStates;

*Description*

Indicates low-priority information that might not be important to the user.

d) *ToString*

[C#]	public	const	AccessibleStates	AlertMedium;
[C++]	public:	const	AccessibleStates	AlertMedium;
[VB]	Public	Const	AlertMedium	As AccessibleStates
[JScript]	public	var	AlertMedium	: AccessibleStates;

*Description*

Indicates important information that does not need to be conveyed to the user immediately. For example, when a battery level indicator is starting to reach a low level, it could generate a medium-level alert. Blind access utilities could then generate a sound to let the user know that important information is available, without actually interrupting the user's work. The user can then query the alert information at his or her leisure.

e) *ToString*

[C#]	public	const	AccessibleStates	Animated;
[C++]	public:	const	AccessibleStates	Animated;

1 [VB] Public Const Animated As AccessibleStates

2 [JScript] public var Animated : AccessibleStates;

3  
4 *Description*

5 The object has a rapidly or constantly changing appearance. Graphics that are  
6 occasionally animated, but not always, should be defined as  
7 **System.Windows.Forms.AccessibleRole.Graphic OR Animated** . This state  
8 should not be used to indicate that the object's location is changing.

9  
10 *f) ToString*

11 [C#] public const AccessibleStates Busy;

12 [C++] public: const AccessibleStates Busy;

13 [VB] Public Const Busy As AccessibleStates

14 [JScript] public var Busy : AccessibleStates;

15  
16 *Description*

17 A control that cannot accept input in its current condition.

18  
19 *g) ToString*

20 [C#] public const AccessibleStates Checked;

21 [C++] public: const AccessibleStates Checked;

22 [VB] Public Const Checked As AccessibleStates

23 [JScript] public var Checked : AccessibleStates;

24  
25 *Description*

An object with a selected check box.

h) ToString

[C#]	public	const	AccessibleStates	Collapsed;
[C++]	public:	const	AccessibleStates	Collapsed;
[VB]	Public	Const	Collapsed	As AccessibleStates
[JScript]	public	var	Collapsed	: AccessibleStates;

Description

The children of the object that are items in an outline or tree structure are hidden.

i) ToString

[C#]	public	const	AccessibleStates	Default;
[C++]	public:	const	AccessibleStates	Default;
[VB]	Public	Const	Default	As AccessibleStates
[JScript]	public	var	Default	: AccessibleStates;

Description

The default button or menu item.

j) ToString

[C#]	public	const	AccessibleStates	Expanded;
[C++]	public:	const	AccessibleStates	Expanded;
[VB]	Public	Const	Expanded	As AccessibleStates
[JScript]	public	var	Expanded	: AccessibleStates;

## Description

The children of the object that are items in an outline or tree structure are displayed.

### k) ToString

[C#]	public	const	AccessibleStates	ExtSelectable;
[C++]	public:	const	AccessibleStates	ExtSelectable;
[VB]	Public	Const	ExtSelectable	As AccessibleStates
[JScript]	public	var	ExtSelectable	: AccessibleStates;

## Description

Alters the selection so that all objects between the selection anchor, which is the object with the keyboard focus, and this object take on the anchor object's selection state. If the anchor object is not selected, the objects are removed from the selection. If the anchor object is selected, the selection is extended to include this object and all the objects in between. You can set the selection state by combining this with

**System.Windows.Forms.AccessibleSelection.AddSelection** or **System.Windows.Forms.AccessibleSelection.RemoveSelection** . This tate does not change the focus or the selection anchor unless it is combined with **System.Windows.Forms.AccessibleSelection.TakeFocus** .

### l) ToString

[C#]	public	const	AccessibleStates	Floating;
[C++]	public:	const	AccessibleStates	Floating;
[VB]	Public	Const	Floating	As AccessibleStates
[JScript]	public	var	Floating	: AccessibleStates;

*Description*

The object is not fixed to the boundary of its parent object, and does not move automatically along with the parent.

*m) ToString*

[C#]	public	const	AccessibleStates	Focusable;
[C++]	public:	const	AccessibleStates	Focusable;
[VB]	Public	Const	Focusable	As AccessibleStates
[JScript]	public	var	Focusable	: AccessibleStates;

*Description*

The object is on the active window and can receive keyboard focus.

*n) ToString*

[C#]	public	const	AccessibleStates	Focused;
[C++]	public:	const	AccessibleStates	Focused;
[VB]	Public	Const	Focused	As AccessibleStates
[JScript]	public	var	Focused	: AccessibleStates;

*Description*

An object with the keyboard focus.

*o) ToString*

[C#]	public	const	AccessibleStates	HotTracked;
------	--------	-------	------------------	-------------

[C++]	public:	const	AccessibleStates	HotTracked;
[VB]	Public	Const	HotTracked	As AccessibleStates
[JScript]	public	var	HotTracked	: AccessibleStates;

*Description*

The object is hot-tracked by the mouse, meaning its appearance is highlighted to indicate the mouse pointer is located over it.

*p) ToString*

[C#]	public	const	AccessibleStates	Indeterminate;
[C++]	public:	const	AccessibleStates	Indeterminate;
[VB]	Public	Const	Indeterminate	As AccessibleStates
[JScript]	public	var	Indeterminate	: AccessibleStates;

*Description*

A three-state check box or toolbar button whose state is indeterminate. The check box is neither checked nor unchecked and it is in the third or mixed state.

*q) ToString*

[C#]	public	const	AccessibleStates	Invisible;
[C++]	public:	const	AccessibleStates	Invisible;
[VB]	Public	Const	Invisible	As AccessibleStates
[JScript]	public	var	Invisible	: AccessibleStates;

*Description*

An object without a visible user interface.

**r) ToString**

[C#]	public	const	AccessibleStates	Linked;
[C++]	public:	const	AccessibleStates	Linked;
[VB]	Public	Const	Linked	As AccessibleStates
[JScript]	public	var	Linked	: AccessibleStates;

*Description*

A linked object that has not been previously selected.

**s) ToString**

[C#]	public	const	AccessibleStates	Marqueed;
[C++]	public:	const	AccessibleStates	Marqueed;
[VB]	Public	Const	Marqueed	As AccessibleStates
[JScript]	public	var	Marqueed	: AccessibleStates;

*Description*

An object with scrolling or moving text or graphics.

**t) ToString**

[C#]	public	const	AccessibleStates	Mixed;
[C++]	public:	const	AccessibleStates	Mixed;
[VB]	Public	Const	Mixed	As AccessibleStates
[JScript]	public	var	Mixed	: AccessibleStates;



*Description*

A three-state check box or toolbar button whose state is indeterminate. The check box is neither checked nor unchecked and it is in the third or mixed state.

*u) ToString*

[C#]	public	const	AccessibleStates	Moveable;
[C++]	public:	const	AccessibleStates	Moveable;
[VB]	Public	Const	Moveable	As AccessibleStates
[JScript]	public	var	Moveable	: AccessibleStates;

*Description*

A movable object.

*v) ToString*

[C#]	public	const	AccessibleStates	MultiSelectable;
[C++]	public:	const	AccessibleStates	MultiSelectable;
[VB]	Public	Const	MultiSelectable	As AccessibleStates
[JScript]	public	var	MultiSelectable	: AccessibleStates;

*Description*

An object that accepts multiple selected items.

*w) ToString*

[C#]	public	const	AccessibleStates	None;
------	--------	-------	------------------	-------

[C++]	public:	const	AccessibleStates	None;
[VB]	Public	Const	None	As AccessibleStates
[JScript]	public	var	None	: AccessibleStates;

*Description*

No state.

*x) ToString*

[C#]	public	const	AccessibleStates	Offscreen;
[C++]	public:	const	AccessibleStates	Offscreen;
[VB]	Public	Const	Offscreen	As AccessibleStates
[JScript]	public	var	Offscreen	: AccessibleStates;

*Description*

No on-screen representation. A sound or alert object would have this state, or a hidden window that is never made visible.

*y) ToString*

[C#]	public	const	AccessibleStates	Pressed;
[C++]	public:	const	AccessibleStates	Pressed;
[VB]	Public	Const	Pressed	As AccessibleStates
[JScript]	public	var	Pressed	: AccessibleStates;

*Description*

A pressed object.

z) *ToString*

[C#]	public	const	AccessibleStates	Protected;
[C++]	public:	const	AccessibleStates	Protected;
[VB]	Public	Const	Protected	As AccessibleStates
[JScript]	public	var	Protected	: AccessibleStates;

*Description*

A password-protected edit control.

aa) *ToString*

[C#]	public	const	AccessibleStates	ReadOnly;
[C++]	public:	const	AccessibleStates	ReadOnly;
[VB]	Public	Const	ReadOnly	As AccessibleStates
[JScript]	public	var	ReadOnly	: AccessibleStates;

*Description*

A read-only object.

bb) *ToString*

[C#]	public	const	AccessibleStates	Selectable;
[C++]	public:	const	AccessibleStates	Selectable;
[VB]	Public	Const	Selectable	As AccessibleStates
[JScript]	public	var	Selectable	: AccessibleStates;

## Description

An object that can accept selection.

### cc) ToString

[C#]            public            const            AccessibleStates            Selected;

[C++]           public:            const            AccessibleStates            Selected;

[VB]           Public            Const            Selected            As            AccessibleStates

[JScript]       public            var            Selected            :            AccessibleStates;

## Description

A selected object.

### dd) ToString

[C#]            public            const            AccessibleStates            SelfVoicing;

[C++]           public:            const            AccessibleStates            SelfVoicing;

[VB]           Public            Const            SelfVoicing            As            AccessibleStates

[JScript]       public            var            SelfVoicing            :            AccessibleStates;

## Description

The object or child can use text-to-speech (TTS) to describe itself. A speech-based accessibility aid should not announce information when an object with this state has the focus because the object will automatically announce information about itself.

*ee) ToString*

[C#]	public	const	AccessibleStates	Sizeable;
[C++]	public:	const	AccessibleStates	Sizeable;
[VB]	Public	Const	Sizeable As	AccessibleStates
[JScript]	public	var	Sizeable :	AccessibleStates;

*Description*

A sizable object.

*ff) ToString*

[C#]	public	const	AccessibleStates	Traversed;
[C++]	public:	const	AccessibleStates	Traversed;
[VB]	Public	Const	Traversed As	AccessibleStates
[JScript]	public	var	Traversed :	AccessibleStates;

*Description*

A linked object that has previously been selected.

*gg) ToString*

[C#]	public	const	AccessibleStates	Unavailable;
[C++]	public:	const	AccessibleStates	Unavailable;
[VB]	Public	Const	Unavailable As	AccessibleStates
[JScript]	public	var	Unavailable :	AccessibleStates;

*Description*

An unavailable object.

*hh) ToString*

[C#]	public	const	AccessibleStates	Valid;
[C++]	public:	const	AccessibleStates	Valid;
[VB]	Public	Const	Valid	As AccessibleStates
[JScript]	public	var	Valid	: AccessibleStates;

*Description*

A valid object.

AmbientProperties class (System.Windows.Forms)

*a) ToString*

*Description*

Provides ambient property values to top-level controls.

An ambient property is a property on a control that, if not set, is retrieved from the parent control. If the control does not have a parent and the property is not set, the control tries to find the value of the ambient property through the **System.ComponentModel.Component.Site** property. If the control is not sited, the site does not support ambient properties, or if the property is not set on the **System.Windows.Forms.AmbientProperties** object, the **System.Windows.Forms.Control** uses its own default values. Some objects derived from the **System.Windows.Forms.Control** class might set the property even if you do not. For example, the **System.Windows.Forms.Form** class always sets the **System.Windows.Forms.Control.ForeColor** and **System.Windows.Forms.Control.BackColor** properties.

*b) AmbientProperties*

*Example Syntax:*

*c) ToString*

```
[C#]                public                AmbientProperties();
[C++]                public:                AmbientProperties();
[VB]                Public                Sub                New()
[JScript] public function AmbientProperties();
```

*d) BackColor*

*e) ToString*

```
[C#]                public                Color                BackColor                {get;                set;}
[C++]                public: __property Color get_BackColor();public: __property void
set_BackColor(Color);
[VB]                Public                Property                BackColor                As                Color
[JScript] public function get BackColor() : Color;public function set
BackColor(Color);
```

*Description*

Gets or sets the ambient background color of an object.

If there is no ambient background color, the value of the **System.Windows.Forms.Control.BackColor** property is set to **System.Drawing.Color.Empty** .

1        *f)      Cursor*

2        *g)      ToString*

3  
4        [C#]            public            Cursor            Cursor            {get;            set;}

5        [C++]    public:    \_\_property    Cursor\*    get\_Cursor();public:    \_\_property    void  
6        set\_Cursor(Cursor\*);

7        [VB]            Public            Property            Cursor            As            Cursor

8        [JScript] public function get Cursor() : Cursor;public function set Cursor(Cursor);

9  
10        *Description*

11        Gets or sets the ambient cursor of an object.

12        If there is no ambient cursor, the value of the  
13        **System.Windows.Forms.Control.Cursor** property is **null** .

14        *h)      Font*

15        *i)      ToString*

16        [C#]            public            Font            Font            {get;            set;}

17        [C++]    public:    \_\_property    Font\*    get\_Font();public:    \_\_property    void  
18        set\_Font(Font\*);

19        [VB]            Public            Property            Font            As            Font

20        [JScript] public function get Font() : Font;public function set Font(Font);

21  
22        *Description*

23        Gets or sets the ambient font of an object.

24        If there is no ambient font, the value of the  
25        **System.Windows.Forms.Control.Font** property is **null** .



j) *ForeColor*

k) *ToString*

[C#]            public            Color            ForeColor            {get;            set;}

[C++]   public:   \_\_property   Color   get\_ForeColor();public:   \_\_property   void  
set\_ForeColor(Color);

[VB]            Public            Property            ForeColor            As            Color

[JScript]   public   function   get   ForeColor()   :   Color;public   function   set  
ForeColor(Color);

#### *Description*

Gets or sets the ambient foreground color of an object.

If there is no ambient foreground color, the value of the  
**System.Windows.Forms.Control.ForeColor** property is set to  
**System.Drawing.Color.Empty** .

AnchorStyles enumeration (System.Windows.Forms)

a) *ToString*

#### *Description*

Specifies how a control anchors to the edges of its container.

When a control is anchored to an edge of its container, the distance between the control and the specified edge remains constant when the container resizes. For example, if a control is anchored to the right edge of its container, the distance between the right edge of the control and the right edge of the container remains constant when the container resizes. A control can be anchored to any combination of control edges. If the control is anchored to opposite edges of its container (for example, to the top and bottom), it resizes when the container resizes. If a control has its **System.Windows.Forms.Control.Anchor** property set to **AnchorStyle.None** , the control moves half of the distance that the

container of the control is resized. For example, if a **System.Windows.Forms.Button** has its **System.Windows.Forms.Control.Anchor** property set to **AnchorStyle.None** and the **System.Windows.Forms.Form** that the control is located on is resized by 20 pixels in either direction, the button will be moved 10 pixels in both directions.

*b) ToString*

[C#]	public	const	AnchorStyles	Bottom;
[C++]	public:	const	AnchorStyles	Bottom;
[VB]	Public	Const	Bottom	As AnchorStyles
[JScript]	public	var	Bottom	: AnchorStyles;

*Description*

The control is anchored to the bottom edge of its container.

*c) ToString*

[C#]	public	const	AnchorStyles	Left;
[C++]	public:	const	AnchorStyles	Left;
[VB]	Public	Const	Left	As AnchorStyles
[JScript]	public	var	Left	: AnchorStyles;

*Description*

The control is anchored to the left edge of its container.

*d) ToString*

[C#]	public	const	AnchorStyles	None;
------	--------	-------	--------------	-------

[C++]	public:	const	AnchorStyles	None;
[VB]	Public	Const	None	As AnchorStyles
[JScript]	public	var	None	: AnchorStyles;

*Description*

The control is not anchored to any edges of its container.

*e) ToString*

[C#]	public	const	AnchorStyles	Right;
[C++]	public:	const	AnchorStyles	Right;
[VB]	Public	Const	Right	As AnchorStyles
[JScript]	public	var	Right	: AnchorStyles;

*Description*

The control is anchored to the right edge of its container.

*f) ToString*

[C#]	public	const	AnchorStyles	Top;
[C++]	public:	const	AnchorStyles	Top;
[VB]	Public	Const	Top	As AnchorStyles
[JScript]	public	var	Top	: AnchorStyles;

*Description*

The control is anchored to the top edge of its container.

Appearance enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the appearance of a control.

Use the members of this enumeration in controls that provide the **System.Windows.Forms.CheckBox.Appearance** property to set its value.

*b) ToString*

[C#]	public	const	Appearance	Button;
[C++]	public:	const	Appearance	Button;
[VB]	Public	Const	Button	As Appearance
[JScript]	public	var	Button	: Appearance;

*Description*

The appearance of a Windows button.

*c) ToString*

[C#]	public	const	Appearance	Normal;
[C++]	public:	const	Appearance	Normal;
[VB]	Public	Const	Normal	As Appearance
[JScript]	public	var	Normal	: Appearance;

*Description*

The default appearance defined by the control class.

Application class (System.Windows.Forms)

a) *ToString*

### *Description*

Provides **static** methods and properties to manage an application, such as methods to start and stop an application, to process Windows messages, and properties to get information about an application. This class cannot be inherited.

The **System.Windows.Forms.Application** class has methods to start and stop applications and threads, and to process Windows messages. Call **System.Windows.Forms.Application.Run** to start an application message loop on the current thread and, optionally, to make a form visible. Call **System.Windows.Forms.Application.Exit** or **System.Windows.Forms.Application.ExitThread** to stop a message loop. Call **System.Windows.Forms.Application.DoEvents** to process messages while your program is in a loop. Call **System.Windows.Forms.Application.AddMessageFilter(System.Windows.Forms IMessageFilter)** to add a message filter to the application message pump to monitor Windows messages. An **System.Windows.Forms.IMessageFilter** lets you stop an event from being raised or perform special operations before invoking an event handler.

b) *AllowQuit*

c) *ToString*

```
[C#]      public      static      bool      AllowQuit      {get;}
[C++]     public:     __property      static      bool      get_AllowQuit();
[VB]      Public      Shared      ReadOnly      Property      AllowQuit      As      Boolean
[JScript] public      static      function      get      AllowQuit()      :      Boolean;
```

### *Description*

Gets a value indicating whether the caller can quit this application.

This property returns **false** if it is called from a **System.Windows.Forms.Control** being hosted within a Web browser. Thus, the **System.Windows.Forms.Control** cannot quit the **System.Windows.Forms.Application**.

d) *CommonAppDataPath*

e) *ToString*

```
[C#]      public      static      string      CommonAppDataPath      {get;}
[C++]     public:     __property static String* get_CommonAppDataPath();
[VB]      Public Shared ReadOnly Property CommonAppDataPath As String
[JScript] public static function get CommonAppDataPath() : String;
```

#### *Description*

Gets the path for the application data that is shared among all users.

If a path does not exist, one is created in the following format: Base Path\  
**System.Windows.Forms.Application.CompanyName** \  
**System.Windows.Forms.Application.ProductName** \  
**System.Windows.Forms.Application.ProductVersion** Gets the path for the application data that is shared among all users.

f) *CommonAppDataRegistry*

g) *ToString*

```
[C#]      public      static      RegistryKey      CommonAppDataRegistry      {get;}
[C++]     public:     __property static RegistryKey* get_CommonAppDataRegistry();
[VB]      Public Shared ReadOnly Property CommonAppDataRegistry As
RegistryKey
[JScript] public static function get CommonAppDataRegistry() : RegistryKey;
```

## Description

Gets the registry key for the application data that is shared among all users.

Information can be written to the

**System.Windows.Forms.Application.CommonAppDataRegistry** only at install time.

*h)      **CompanyName***

*i)      **ToString***

[C#]      public      static      string      CompanyName      {get;}

[C++]      public:      \_\_property      static      String\*      get\_CompanyName();

[VB]      Public      Shared      ReadOnly      Property      CompanyName      As      String

[JScript]      public      static      function      get      CompanyName()      :      String;

## Description

Gets the company name associated with the application.

*j)      **CurrentCulture***

*k)      **ToString***

[C#]      public      static      CultureInfo      CurrentCulture      {get;      set;}

[C++]      public:      \_\_property      static      CultureInfo\*      get\_CurrentCulture();public:

\_\_property      static      void      set\_CurrentCulture(CultureInfo\*);

[VB]      Public      Shared      Property      CurrentCulture      As      CultureInfo

[JScript]      public      static      function      get      CurrentCulture()      :      CultureInfo;public      static

function      set      CurrentCulture(CultureInfo);

## Description

Gets or sets the culture information for the current thread.

*l) CurrentInputLanguage*

*m) ToString*

```
[C#] public static InputLanguage CurrentInputLanguage {get; set;}
```

```
[C++] public: __property static InputLanguage*
```

```
get_CurrentInputLanguage();public: __property static void
```

```
set_CurrentInputLanguage(InputLanguage*);
```

```
[VB] Public Shared Property CurrentInputLanguage As InputLanguage
```

```
[JScript] public static function get CurrentInputLanguage() :
```

```
InputLanguage;public static function set CurrentInputLanguage(InputLanguage);
```

## Description

Gets or sets the current input language for the current thread.

*n) ExecutablePath*

*o) ToString*

```
[C#] public static string ExecutablePath {get;}
```

```
[C++] public: __property static String* get_ExecutablePath();
```

```
[VB] Public Shared ReadOnly Property ExecutablePath As String
```

```
[JScript] public static function get ExecutablePath() : String;
```



## Description

Gets the path for the executable file that started the application.

*p) LocalUserAppDataPath*

*q) ToString*

```
[C#] public static string LocalUserAppDataPath {get;}
```

```
[C++] public: __property static String* get_LocalUserAppDataPath();
```

```
[VB] Public Shared ReadOnly Property LocalUserAppDataPath As String
```

```
[JScript] public static function get LocalUserAppDataPath() : String;
```

## Description

Gets the path for the application data of a local, non-roaming user.

A local user is one whose user profile is stored on the system on which the user logged on. If a path does not exist, one is created in the following format: Base Path\ **System.Windows.Forms.Application.CompanyName** \ **System.Windows.Forms.Application.ProductName** \ **System.Windows.Forms.Application.ProductVersion** Gets the path for the application data specific to a local, non-roaming user.

*r) MessageLoop*

*s) ToString*

```
[C#] public static bool MessageLoop {get;}
```

```
[C++] public: __property static bool get_MessageLoop();
```

```
[VB] Public Shared ReadOnly Property MessageLoop As Boolean
```

```
[JScript] public static function get MessageLoop() : Boolean;
```

## Description

Gets a value indicating whether a message loop exists on this thread.

t) **ProductName**

u) **ToString**

```
[C#]      public      static      string      ProductName      {get;}
```

```
[C++]     public:     __property     static     String*     get_ProductName();
```

```
[VB]      Public      Shared      ReadOnly      Property      ProductName      As      String
```

```
[JScript] public      static      function      get      ProductName()      :      String;
```

## Description

Gets the product name associated with this application.

v) **ProductVersion**

w) **ToString**

```
[C#]      public      static      string      ProductVersion      {get;}
```

```
[C++]     public:     __property     static     String*     get_ProductVersion();
```

```
[VB]      Public      Shared      ReadOnly      Property      ProductVersion      As      String
```

```
[JScript] public      static      function      get      ProductVersion()      :      String;
```

## Description

Gets the product version associated with this application.

Typically, a version number displays as "major number.minor number.build number.private part number".

x) *SafeTopLevelCaptionFormat*

y) *ToString*

```
[C#] public static string SafeTopLevelCaptionFormat {get; set;}
[C++] public: __property static String* get_SafeTopLevelCaptionFormat();public:
__property static void set_SafeTopLevelCaptionFormat(String*);
[VB] Public Shared Property SafeTopLevelCaptionFormat As String
[JScript] public static function get SafeTopLevelCaptionFormat() : String;public
static function set SafeTopLevelCaptionFormat(String);
```

*Description*

Gets or sets the format string to apply to top-level window captions when they are displayed with a warning banner.

z) *StartupPath*

aa) *ToString*

```
[C#] public static string StartupPath {get;}
[C++] public: __property static String* get_StartupPath();
[VB] Public Shared ReadOnly Property StartupPath As String
[JScript] public static function get StartupPath() : String;
```

*Description*

Gets the path for the executable file that started the application.

1       **bb)    UserAppDataPath**

2       **cc)    ToString**

3  
4   [C#]       public       static       string       UserAppDataPath       {get;}  
5   [C++]    public:    \_\_property   static   String\*   get\_UserAppDataPath();  
6   [VB]   Public   Shared   ReadOnly   Property   UserAppDataPath   As   String  
7   [JScript]   public   static   function   get   UserAppDataPath()   :   String;

8  
9       *Description*

10   Gets the path for the application data of a roaming user.

11   If a path does not exist, one is created in the following format: Base Path\  
12   **System.Windows.Forms.Application.CompanyName \**  
13   **System.Windows.Forms.Application.ProductName \**  
14   **System.Windows.Forms.Application.ProductVersion** A roaming user works  
15   on more than one computer in a network. The user profile for a roaming user is  
16   kept on a server on the network and is loaded onto a system when the user logs  
17   on.

15       **dd)    UserAppDataRegistry**

16       **ee)    ToString**

17  
18   [C#]       public       static       RegistryKey    UserAppDataRegistry    {get;}  
19   [C++]    public:    \_\_property   static   RegistryKey\*   get\_UserAppDataRegistry();  
20   [VB]   Public   Shared   ReadOnly   Property   UserAppDataRegistry   As   RegistryKey  
21   [JScript]   public   static   function   get   UserAppDataRegistry()   :   RegistryKey;

22  
23       *Description*

24   Gets the registry key of the application data specific to the roaming user.  
25

Information can be written into the **System.Windows.Forms.Application.UserAppDataRegistry** only at install time.

*ff) ToString*

[C#]      public      static      event      EventHandler      ApplicationExit;

[C++]      public:      static      \_\_event      EventHandler\*      ApplicationExit;

[VB]      Public      Shared      Event      ApplicationExit      As      EventHandler

*Description*

Occurs when the application is about to shut down.

You must attach the event handlers to the **System.Windows.Forms.Application.Exit** event to perform unhandled, required tasks before the application stops running. You can close files opened by this application, or dispose of objects that garbage collection did not reclaim.

*gg) ToString*

[C#]      public      static      event      EventHandler      Idle;

[C++]      public:      static      \_\_event      EventHandler\*      Idle;

[VB]      Public      Shared      Event      Idle      As      EventHandler

*Description*

Occurs when the application finishes processing and is about to enter the idle state.

If you have tasks that you must perform before the thread becomes idle, attach them to this event.

## hh) ToString

```
[C#] public static event ThreadExceptionHandler ThreadException;  
[C++] public: static __event ThreadExceptionHandler* ThreadException;  
[VB] Public Shared Event ThreadException As ThreadExceptionHandler
```

### Description

Occurs when an untrapped thread exception is thrown.

This event enables an application to handle an exception intelligently when it receives a thread exception from a window procedure. Attach your event handlers to the **System.Windows.Forms.Application.ThreadException** event to deal with the exception. An appropriate event handler does not terminate the thread, and allow your application to continue executing.

## ii) ToString

```
[C#] public static event EventHandler ThreadExit;  
[C++] public: static __event EventHandler* ThreadExit;  
[VB] Public Shared Event ThreadExit As EventHandler
```

### Description

Occurs when a thread is about to shut down. When the main thread for an application is about to be shut down, this event is raised first, followed by an **System.Windows.Forms.Application.ApplicationExit** event.

You must attach the event handlers to the **System.Windows.Forms.Application.ThreadExit** event to perform any unhandled, required tasks before the thread stops running. Close files opened by this thread, or dispose of objects that the garbage collector did not reclaim.

### jj) AddMessageFilter

```
[C#]    public    static    void    AddMessageFilter(IMessageFilter    value);  
[C++]   public:   static    void    AddMessageFilter(IMessageFilter*    value);  
[VB]    Public    Shared    Sub    AddMessageFilter(ByVal    value    As    IMessageFilter)  
[JScript] public static function AddMessageFilter(value : IMessageFilter);
```

#### Description

Adds a message filter to monitor Windows messages as they are routed to their destinations.

Use a message filter to prevent specific events from being raised or to perform special operations for an event before it is passed to an event handler. Message filters are unique to a specific thread. The implementation of the **System.Windows.Forms.IMessageFilter** interface you want to install.

### kk) DoEvents

```
[C#]          public          static          void          DoEvents();  
[C++]         public:         static          void          DoEvents();  
[VB]          Public          Shared          Sub          DoEvents()  
[JScript]     public          static          function      DoEvents();
```

#### Description

Processes all Windows messages currently in the message queue.

When you run a Windows Form, it creates the new form, which then waits for events to handle. Each time the form handles an event, it processes all the code associated with that event. All other events wait in the queue. While your code handles the event, your application does not respond. For example, the window does not repaint if another window is dragged on top.

## ll) *Exit*

[C#]	public	static	void	Exit();
[C++]	public:	static	void	Exit();
[VB]	Public	Shared	Sub	Exit()
[JScript]	public	static	function	Exit();

### *Description*

Informs all message pumps that they must terminate, and then closes all application windows after the messages have been processed.

This method stops all running message loops on all threads and closes all windows of the application. This method does not actually force the application to exit, but rather causes all calls to

**System.Windows.Forms.Application.Run** to return. To exit a message loop for the current thread only, call

**System.Windows.Forms.Application.ExitThread** .

## mm) *ExitThread*

[C#]	public	static	void	ExitThread();
[C++]	public:	static	void	ExitThread();
[VB]	Public	Shared	Sub	ExitThread()
[JScript]	public	static	function	ExitThread();

### *Description*

Exits the message loop on the current thread and closes all windows on the thread.

Use this method to exit the message loop of the current thread. This method causes the call to **System.Windows.Forms.Application.Run** for the current thread to return. To exit the entire application, call

**System.Windows.Forms.Application.Exit** .



*nn) OleRequired*

```
[C#]      public      static      ApartmentState      OleRequired();
[C++]      public:      static      ApartmentState      OleRequired();
[VB]      Public      Shared      Function      OleRequired()      As      ApartmentState
[JScript]      public      static      function      OleRequired()      :      ApartmentState;
```

*Description*

Initializes OLE on the current thread.

*Return Value:* One of the **System.Threading.ApartmentState** values.

Call this method before calling any **Microsoft.Win32** method that requires OLE.

*oo) OnThreadException*

```
[C#]      public      static      void      OnThreadException(Exception      t);
[C++]      public:      static      void      OnThreadException(Exception*      t);
[VB]      Public      Shared      Sub      OnThreadException(ByVal t As Exception)
[JScript]      public      static      function      OnThreadException(t : Exception);
```

*Description*

Raises the **System.Windows.Forms.Application.ThreadException** event.

This event enables an application to handle an exception intelligently. A window procedure calls this event when it receives a thread exception. Attach your event handlers to this event. An **System.Exception** that represents the exception that was thrown.

*pp) RemoveMessageFilter*

```
[C#]      public      static      void      RemoveMessageFilter(IMessageFilter      value);
```

```

1 [C++] public: static void RemoveMessageFilter(IMessageFilter* value);
2 [VB] Public Shared Sub RemoveMessageFilter(ByVal value As IMessageFilter)
3 [JScript] public static function RemoveMessageFilter(value : IMessageFilter);

```

#### 5 *Description*

6 Removes a message filter from the message pump of the application.

7 You can remove a message filter when you no longer want to capture Windows  
8 messages before they are dispatched. The implementation of the  
**System.Windows.Forms.IMessageFilter** to remove from the application.

#### 9 *qq) Run*

```

10
11 [C#]          public          static          void          Run();
12 [C++]          public:          static          void          Run();
13 [VB]          Public          Shared          Sub          Run()
14 [JScript] public static function Run(); Begins running a standard application
15 message          loop          on          the          current          thread.

```

#### 17 *Description*

18 Begins running a standard application message loop on the current thread,  
19 without a form.

20 The message loop runs until **System.Windows.Forms.Application.Exit** or  
**System.Windows.Forms.Application.ExitThread** is called.

#### 21 *rr) Run*

```

22
23 [C#]          public          static          void          Run(ApplicationContext context);
24 [C++]          public:          static          void          Run(ApplicationContext* context);

```

1 [VB] Public Shared Sub Run(ByVal context As ApplicationContext)

2 [JScript] public static function Run(context : ApplicationContext);

3  
4 *Description*

5 Begins running a standard application message loop on the current thread, with  
an **System.Windows.Forms.ApplicationContext** .

6 The message loop runs until **System.Windows.Forms.Application.Exit** or  
7 **System.Windows.Forms.Application.ExitThread** is called or the  
8 **System.Windows.Forms.Application.ThreadExit** event is raised on the  
context object. An **System.Windows.Forms.ApplicationContext** in which the  
application is run.

9  
10 ss) *Run*

11  
12 [C#] public static void Run(Form mainForm);

13 [C++] public: static void Run(Form\* mainForm);

14 [VB] Public Shared Sub Run(ByVal mainForm As Form)

15 [JScript] public static function Run(mainForm : Form);

16  
17 *Description*

18 Begins running a standard application message loop on the current thread, and  
makes the specified form visible.

19 Typically, the main function of an application calls this method and passes it to  
the main window of the application. A **System.Windows.Forms.Form** that  
20 represents the form to make visible.

ApplicationContext class (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the contextual information about an application thread.

You can use the **System.Windows.Forms.ApplicationContext** class to redefine the circumstances that cause a message loop to exit. By default, the **System.Windows.Forms.ApplicationContext** listens to the **System.Windows.Forms.Form.Closed** event on the application's main **System.Windows.Forms.Form**, then exits the thread's message loop.

*b) ApplicationContext*

*Example Syntax:*

*c) ToString*

[C#]	public	ApplicationContext();
[C++]	public:	ApplicationContext();
[VB]	Public	Sub New()

[JScript] public function ApplicationContext(); Initializes a new instance of the **System.Windows.Forms.ApplicationContext** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.ApplicationContext** class with no context.

*d) ApplicationContext*

*Example Syntax:*

e) *ToString*

```
[C#]      public      ApplicationContext(Form      MainForm);
[C++]     public:      ApplicationContext(Form*      MainForm);
[VB]      Public      Sub      New(ByVal      MainForm      As      Form)
[JScript] public      function      ApplicationContext(MainForm      :      Form);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ApplicationContext** class with the specified **System.Windows.Forms.Form**.

If **System.Windows.Forms.ApplicationContext.OnMainFormClosed(System.Object, System.EventArgs)** is not overridden, the message loop of the thread terminates when **System.Windows.Forms.ApplicationContext.MainForm** is closed. The main **System.Windows.Forms.Form** of the application to use for context.

f) *MainForm*

g) *ToString*

```
[C#]      public      Form      MainForm      {get;      set;}
[C++]     public:      __property      Form*      get_MainForm();public:      __property      void
set_MainForm(Form*);
[VB]      Public      Property      MainForm      As      Form
[JScript] public      function      get      MainForm()      :      Form;public      function      set
MainForm(Form);
```

*Description*

Gets or sets the **System.Windows.Forms.Form** to use as context.

This property determines the main **System.Windows.Forms.Form** for this context. This property can change at any time. If **System.Windows.Forms.ApplicationContext.OnMainFormClosed(System.Object, System.EventArgs)** is not overridden, the message loop of the thread terminates when the *mainForm* parameter closes.

*h) ToString*

[C#]            public            event            EventHandler            ThreadExit;

[C++]           public:            \_\_event            EventHandler\*            ThreadExit;

[VB]            Public            Event            ThreadExit            As            EventHandler

*Description*

Occurs when the message loop of the thread should be terminated, by calling **System.Windows.Forms.ApplicationContext.ExitThread** .

*i) Dispose*

[C#]                            public                            void                            Dispose();

[C++]                            public:                            void                            Dispose();

[VB]                            Public                            Sub                            Dispose()

[JScript] public function Dispose(); Releases the resources used by the **System.Windows.Forms.ApplicationContext** .

*Description*

Releases all resources used by the **System.Windows.Forms.ApplicationContext** .

Calling **System.Windows.Forms.ApplicationContext.Dispose** allows the resources used by the **System.Windows.Forms.ApplicationContext** . to be

reallocated for other purposes. For more information about **System.Windows.Forms.ApplicationContext.Dispose** , see .

*j) Dispose*

[C#]	protected	virtual	void	Dispose(bool disposing);
[C++]	protected:	virtual	void	Dispose(bool disposing);
[VB]	Overridable	Protected	Sub	Dispose(ByVal disposing As Boolean)
[JScript]	protected	function	Dispose(disposing : Boolean);	

*Description*

Releases the unmanaged resources used by the **System.Windows.Forms.ApplicationContext** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

*k) ExitThread*

[C#]	public	void	ExitThread();
[C++]	public:	void	ExitThread();
[VB]	Public	Sub	ExitThread()
[JScript]	public	function	ExitThread();

*Description*

Terminates the message loop of the thread.

This method calls **System.Windows.Forms.ApplicationContext.ExitThreadCore** .

# l) *ExitThreadCore*

[C#]	protected	virtual	void	ExitThreadCore();
[C++]	protected:	virtual	void	ExitThreadCore();
[VB]	Overridable	Protected	Sub	ExitThreadCore()
[JScript]	protected		function	ExitThreadCore();

## *Description*

Terminates the message loop of the thread.

This method is called from

**System.Windows.Forms.ApplicationContext.ExitThread .**

# m) *Finalize*

[C#]				~ApplicationContext();
[C++]				~ApplicationContext();
[VB]	Overrides	Protected	Sub	Finalize()
[JScript]	protected	override	function	Finalize();

## *Description*

# n) *OnMainFormClosed*

[C#]	protected	virtual	void	OnMainFormClosed(object sender, EventArgs e);
[C++]	protected:	virtual	void	OnMainFormClosed(Object* sender, EventArgs* e);
[VB]	Overridable	Protected	Sub	OnMainFormClosed(ByVal sender As Object,



1 ByVal e As EventArgs)

2 [JScript] protected function OnMainFormClosed(sender : Object, e : EventArgs);

3  
4 *Description*

5 Calls **System.Windows.Forms.ApplicationContext.ExitThreadCore** , which  
6 raises the **System.Windows.Forms.ApplicationContext.ThreadExit** event.

7 The default implementation of this method calls  
8 **System.Windows.Forms.ApplicationContext.ExitThreadCore** . The object  
9 that raised the event. The **System.EventArgs** that contains the event data.

10 ArrangeDirection enumeration (System.Windows.Forms)

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
*a) ToString*

*Description*

Specifies the direction in which the system arranges minimized windows.

This enumeration is used by the  
**System.Windows.Forms.SystemInformation.ArrangeDirection** property  
of the **System.Windows.Forms.SystemInformation** class.

*b) ToString*

[C#] public const ArrangeDirection Down;

[C++] public: const ArrangeDirection Down;

[VB] Public Const Down As ArrangeDirection

[JScript] public var Down : ArrangeDirection;

*Description*

Arranged vertically, from top to bottom.

c) *ToString*

[C#]	public	const	ArrangeDirection	Left;
[C++]	public:	const	ArrangeDirection	Left;
[VB]	Public	Const	Left As	ArrangeDirection
[JScript]	public	var	Left :	ArrangeDirection;

*Description*

Arranged horizontally, from left to right.

d) *ToString*

[C#]	public	const	ArrangeDirection	Right;
[C++]	public:	const	ArrangeDirection	Right;
[VB]	Public	Const	Right As	ArrangeDirection
[JScript]	public	var	Right :	ArrangeDirection;

*Description*

Arranged horizontally, from right to left.

e) *ToString*

[C#]	public	const	ArrangeDirection	Up;
[C++]	public:	const	ArrangeDirection	Up;
[VB]	Public	Const	Up As	ArrangeDirection
[JScript]	public	var	Up :	ArrangeDirection;

1  
2 *Description*

3 Arranged vertically, from bottom to top.

4 ArrangeStartingPosition enumeration (System.Windows.Forms)

5 *a) ToString*

6  
7  
8 *Description*

9 Specifies the starting position that the system uses to arrange minimized windows.

10 This enumeration is used by the  
11 **System.Windows.Forms.SystemInformation.ArrangeStartingPosition**  
12 property of the **System.Windows.Forms.SystemInformation** class.

13 *b) ToString*

14 [C#] public const ArrangeStartingPosition BottomLeft;

15 [C++] public: const ArrangeStartingPosition BottomLeft;

16 [VB] Public Const BottomLeft As ArrangeStartingPosition

17 [JScript] public var BottomLeft : ArrangeStartingPosition;

18  
19  
20 *Description*

21 Starts at the lower-left corner of the screen, which is the default position.

22 *c) ToString*

23 [C#] public const ArrangeStartingPosition BottomRight;

24 [C++] public: const ArrangeStartingPosition BottomRight;

1 [VB] Public Const BottomRight As ArrangeStartingPosition

2 [JScript] public var BottomRight : ArrangeStartingPosition;

3  
4 *Description*

5 Starts at the lower-right corner of the screen.

6 *d) ToString*

7  
8 [C#] public const ArrangeStartingPosition Hide;

9 [C++] public: const ArrangeStartingPosition Hide;

10 [VB] Public Const Hide As ArrangeStartingPosition

11 [JScript] public var Hide : ArrangeStartingPosition;

12  
13 *Description*

14 Hides minimized windows by moving them off the visible area of the screen.

15 *e) ToString*

16  
17 [C#] public const ArrangeStartingPosition TopLeft;

18 [C++] public: const ArrangeStartingPosition TopLeft;

19 [VB] Public Const TopLeft As ArrangeStartingPosition

20 [JScript] public var TopLeft : ArrangeStartingPosition;

21  
22 *Description*

23 Starts at the upper-left corner of the screen.

*f) ToString*

```
[C#]      public      const      ArrangeStartingPosition      TopRight;
[C++]     public:     const      ArrangeStartingPosition      TopRight;
[VB]      Public      Const      TopRight      As      ArrangeStartingPosition
[JScript] public      var      TopRight      :      ArrangeStartingPosition;
```

*Description*

Starts at the upper-right corner of the screen.

AxHost.AxComponentEditor class (System.Windows.Forms)

*a) ToString*

*b) AxHost.AxComponentEditor*

*Example Syntax:*

*c) ToString*

*d) EditComponent*

```
[C#] public override bool EditComponent(ITypeDescriptorContext context, object
obj, IWin32Window parent);
[C++] public: bool EditComponent(ITypeDescriptorContext* context, Object*
obj, IWin32Window* parent);
[VB] Overrides Public Function EditComponent(ByVal context As
ITypeDescriptorContext, ByVal obj As Object, ByVal parent As IWin32Window)
As Boolean
[JScript] public override function EditComponent(context :
ITypeDescriptorContext, obj : Object, parent : IWin32Window) : Boolean;
```

AxHost class (System.Windows.Forms)

*a) ToString*

*Description*

Wraps ActiveX controls and exposes them as fully featured Windows Forms controls.

You typically do not use the **System.Windows.Forms.AxHost** class directly. You should inherit from this class and add the functionality required by your ActiveX control. You can use the ActiveX Control Importer tool to generate the wrappers that extend **System.Windows.Forms.AxHost** . For more information about the ActiveX Control Importer tool, see .

*b) AxHost*

*Example Syntax:*

*c) ToString*

[C#]                      protected                      AxHost(string                      clsid);

[C++]                      protected:                      AxHost(String\*                      clsid);

[VB]              Protected              Sub              New(ByVal              clsid              As              String)

[JScript] protected function AxHost(clsid : String); Creates a new instance of a control which wraps an activeX control given by the clsid parameter and flags of 0.

*d) AxHost*

*Example Syntax:*

e) *ToString*

[C#]       protected       AxHost(string       clsid,       int       flags);  
[C++]       protected:       AxHost(String\*       clsid,       int       flags);  
[VB] Protected Sub New(ByVal clsid As String, ByVal flags As Integer)  
[JScript] protected function AxHost(clsid : String, flags : int); Creates a new  
instance of a control which wraps an activeX control given by the clsid and flags  
parameters.

1	<i>f) AccessibilityObject</i>
2	<i>g) AccessibleDefaultActionDescription</i>
3	<i>h) AccessibleDescription</i>
4	<i>i) AccessibleName</i>
5	<i>j) AccessibleRole</i>
6	<i>k) AllowDrop</i>
7	<i>l) Anchor</i>
8	<i>m) BackColor</i>
9	<i>n) ToString</i>
10	<i>o) BackgroundImage</i>
11	<i>p) ToString</i>
12	<i>q) BindingContext</i>
13	<i>r) Bottom</i>
14	<i>s) Bounds</i>
15	<i>t) CanFocus</i>
16	<i>u) CanSelect</i>
17	<i>v) Capture</i>
18	<i>w) CausesValidation</i>
19	<i>x) ClientRectangle</i>
20	<i>y) ClientSize</i>
21	<i>z) CompanyName</i>
22	<i>aa) Container</i>
23	<i>bb) ContainingControl</i>
24	
25	



*cc) ToString*

1

2

3

*Description*

4

Gets or sets the parent container of the ActiveX control.

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

*dd) ContainsFocus*  
*ee) ContextMenu*  
*ff) ToString*  
*gg) Controls*  
*hh) Created*  
*ii) CreateParams*  
*jj) ToString*  
*kk) Cursor*  
*ll) ToString*  
*mm) DataBindings*  
*nn) DefaultImeMode*  
*oo) DefaultSize*  
*pp) ToString*  
*qq) DesignMode*  
*rr) DisplayRectangle*  
*ss) Disposing*  
*tt) Dock*  
*uu) EditMode*  
*vv) ToString*  
*ww) Enabled*  
*xx) ToString*

*Description*

Gets or sets a value indicating whether the ActiveX control is in an enabled state.

*yy) Events*

*zz) Focused*

*aaa) Font*

*bbb) ToString*

*ccc) FontHeight*

*ddd) ForeColor*

*eee) ToString*

*fff) Handle*

*ggg) HasAboutBox*

*hhh) ToString*

*Description*

Gets a value indicating whether the ActiveX control has an about dialog box.

The about dialog box typically displays version and copyright information about the ActiveX control.

iii) *HasChildren*  
 jjj) *Height*  
 kkk) *ImeMode*  
 ll) *InvokeRequired*  
 mmm) *IsAccessible*  
 nnn) *IsDisposed*  
 ooo) *IsHandleCreated*  
 ppp) *Left*  
 qq) *Location*  
 rrr) *Name*  
 sss) *OcxState*  
 ttt) *ToString*

### *Description*

Gets or sets the persisted state of the ActiveX control.

The value of the **System.Windows.Forms.AxHost.OcxState** property is used after the control is created but before it is shown. This method computes the persisted state of the underlying ActiveX control and returns it in the encapsulated **System.Windows.Forms.AxHost.State** object. If the control has been modified since it was last saved to a persisted state, it need to be saved.

1        *uuu) Parent*  
 2        *vvv) ProductName*  
 3        *www) ProductVersion*  
 4        *xxx) RecreatingHandle*  
 5        *yyy) Region*  
 6        *zzz) RenderRightToLeft*  
 7        *aaaa) ResizeRedraw*  
 8        *bbbb) Right*  
 9        *cccc) RightToLeft*  
 10       *dddd) ToString*  
 11       *eeee) RightToLeft*  
 12       *ffff) ShowFocusCues*  
 13       *gggg) ShowKeyboardCues*  
 14       *hhhh) Site*  
 15       *iiii) ToString*  
 16       *jjjj) Size*  
 17       *kkkk) TabIndex*  
 18       *llll) TabStop*  
 19       *mmmm)Tag*  
 20       *nnnn) Text*  
 21       *oooo) ToString*  
 22       *pppp) Top*  
 23       *qqqq) TopLevelControl*

rrrr) *Visible*

ssss) *Width*

tttt) *WindowTarget*

uuuu) *ToString*

#### *Description*

Occurs when the **System.Windows.Forms.Control.BackColor** property of the ActiveX control has changed.

For more information about handling events, see .

vvvv) *ToString*

[C#] public new event EventHandler BackgroundImageChanged;

[C++] public: \_\_event EventHandler\* BackgroundImageChanged;

[VB] Shadows Public Event BackgroundImageChanged As EventHandler

#### *Description*

Occurs when the **System.Windows.Forms.Control.BackgroundImage** property of the ActiveX control has changed.

For more information about handling events, see .

wwwv) *ToString*

[C#] public new event EventHandler BindingContextChanged;

[C++] public: \_\_event EventHandler\* BindingContextChanged;

[VB] Shadows Public Event BindingContextChanged As EventHandler

*Description*

Occurs when the **System.Windows.Forms.BindingContext** property of the ActiveX control has changed.

For more information about handling events, see .

*xxxx) ToString*

*Description*

Occurs when focus or keyboard or both cues have changed.

For more information about handling events, see .

*yyyy) ToString*

[C#]	public	new	event	EventHandler	Click;
[C++]	public:		__event	EventHandler*	Click;
[VB]	Shadows	Public	Event	Click	As EventHandler

*Description*

Occurs when the control is clicked.

For more information about handling events, see .

*zzzz) ToString*

[C#]	public	new	event	EventHandler	ContextMenuChanged;
[C++]	public:		__event	EventHandler*	ContextMenuChanged;
[VB]	Shadows	Public	Event	ContextMenuChanged	As EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.ContextMenu** property value has changed.

For more information about handling events, see .

*aaaaa) ToString*

*Description*

Occurs when the **System.Windows.Forms.Control.Cursor** property value has changed.

For more information about handling events, see .

*bbbbbb) ToString*

*Description*

Occurs when the control is double clicked.

For more information about handling events, see .

*cccccc) ToString*

```
[C#]      public      new      event      DragEventHandler      DragDrop;
[C++]      public:      __event      DragEventHandler*      DragDrop;
[VB]      Shadows      Public      Event      DragDrop      As      DragEventHandler
```

*Description*

Occurs when a drag-and-drop operation is completed.



For more information about handling events, see .

*dddd) ToString*

```
[C#]      public      new      event      DragEventHandler      DragEnter;
[C++]      public:      __event      DragEventHandler*      DragEnter;
[VB]      Shadows      Public      Event      DragEnter      As      DragEventHandler
```

*Description*

Occurs when an object is dragged into the control's bounds.

For more information about handling events, see .

*eeee) ToString*

```
[C#]      public      new      event      EventHandler      DragLeave;
[C++]      public:      __event      EventHandler*      DragLeave;
[VB]      Shadows      Public      Event      DragLeave      As      EventHandler
```

*Description*

Occurs when an object has been dragged into and out of the control's bounds.

For more information about handling events, see .

*ffff) ToString*

```
[C#]      public      new      event      DragEventHandler      DragOver;
[C++]      public:      __event      DragEventHandler*      DragOver;
[VB]      Shadows      Public      Event      DragOver      As      DragEventHandler
```

## Description

Occurs when an object has been dragged over the control's bounds.

For more information about handling events, see .

### *ggggg) ToString*

[C#]      public      new      event      EventHandler      EnabledChanged;

[C++]      public:      \_\_event      EventHandler\*      EnabledChanged;

[VB]      Shadows      Public      Event      EnabledChanged      As      EventHandler

## Description

Occurs when the **System.Windows.Forms.Control.Enabled** property value has changed.

For more information about handling events, see .

### *hhhhh)ToString*

## Description

Occurs when the **System.Windows.Forms.Control.Font** property value has changed.

For more information about handling events, see .

### *iiii) ToString*

[C#]      public      new      event      EventHandler      ForeColorChanged;

[C++]      public:      \_\_event      EventHandler\*      ForeColorChanged;

[VB]      Shadows      Public      Event      ForeColorChanged      As      EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.ForeColor** property value has changed.

For more information about handling events, see .

*jjjj) ToString*

[C#] public new event GiveFeedbackEventHandler GiveFeedback;

[C++] public: \_\_event GiveFeedbackEventHandler\* GiveFeedback;

[VB] Shadows Public Event GiveFeedback As GiveFeedbackEventHandler

*Description*

Occurs during a drag operation.

The **System.Windows.Forms.Control.GiveFeedback** event allows the source of a drag event to modify the appearance of the mouse pointer in order to give the user visual feedback during a drag-and-drop operation.

*kkkkk) ToString*

*Description*

Occurs when the user requests Help for a control.

For more information about handling events, see .

*lllll) ToString*

[C#] public new event EventHandler ImeModeChanged;

[C++] public: \_\_event EventHandler\* ImeModeChanged;

1 [VB] Shadows Public Event ImeModeChanged As EventHandler

3 *Description*

4 Occurs when the **System.Windows.Forms.Control.ImeMode** property has  
5 changed.

6 For more information about handling events, see .

7 *mmmmm)ToString*

9 *Description*

10 Occurs when a key is pressed down while the control has focus.

11 For more information about handling events, see .

12 *nnnnn)ToString*

14 [C#] public new event KeyPressEventHandler KeyPress;

15 [C++] public: \_\_event KeyPressEventHandler\* KeyPress;

16 [VB] Shadows Public Event KeyPress As KeyPressEventHandler

18 *Description*

19 Occurs when a key is pressed while the control has focus.

20 For more information about handling events, see .

21 *ooooo) ToString*

23 [C#] public new event KeyEventHandler KeyUp;

24 [C++] public: \_\_event KeyEventHandler\* KeyUp;

1 [VB] Shadows Public Event KeyUp As KeyEventHandler

3 *Description*

4 Occurs when a key is released while the control has focus.

5 For more information about handling events, see .

6 *ppppp) ToString*

8 [C#] public new event LayoutEventHandler Layout;

9 [C++] public: \_\_event LayoutEventHandler\* Layout;

10 [VB] Shadows Public Event Layout As LayoutEventHandler

12 *Description*

13 Occurs when a control has to lay out its child controls.

14 For more information about handling events, see .

15 *qqqqq) ToString*

18 *Description*

19 Occurs when the mouse pointer is over the control and a mouse button is pressed.

20 For more information about handling events, see .

21 *rrrrr) ToString*

23 [C#] public new event EventHandler MouseEnter;

24 [C++] public: \_\_event EventHandler\* MouseEnter;

[VB] Shadows Public Event MouseEnter As EventHandler

*Description*

Occurs when the mouse pointer enters the control.

For more information about handling events, see .

*sssss) ToString*

[C#] public new event EventHandler MouseHover;

[C++] public: \_\_event EventHandler\* MouseHover;

[VB] Shadows Public Event MouseHover As EventHandler

*Description*

Occurs when the mouse pointer hovers over the contro.

For more information about handling events, see .

*ttttt) ToString*

[C#] public new event EventHandler MouseLeave;

[C++] public: \_\_event EventHandler\* MouseLeave;

[VB] Shadows Public Event MouseLeave As EventHandler

*Description*

Occurs when the mouse pointer leaves the control.

For more information about handling events, see .

### *uuuuu)ToString*

```
[C#]    public    new    event    MouseEventHandler    MouseMove;
[C++]    public:    __event    MouseEventHandler*    MouseMove;
[VB]    Shadows    Public    Event    MouseMove    As    MouseEventHandler
```

#### *Description*

Occurs when the mouse pointer is moved over the control.

For more information about handling events, see .

### *vvvvv) ToString*

```
[C#]    public    new    event    MouseEventHandler    MouseUp;
[C++]    public:    __event    MouseEventHandler*    MouseUp;
[VB]    Shadows    Public    Event    MouseUp    As    MouseEventHandler
```

#### *Description*

Occurs when the mouse pointer is over the control and a mouse button is released.

For more information about handling events, see .

### *wwwww)ToString*

```
[C#]    public    new    event    MouseEventHandler    MouseWheel;
[C++]    public:    __event    MouseEventHandler*    MouseWheel;
[VB]    Shadows    Public    Event    MouseWheel    As    MouseEventHandler
```

#### *Description*

Occurs when the mouse wheel moves while the control has focus.

For more information about handling events, see .

*xxxxx) ToString*

#### *Description*

Occurs when when the control is redrawn.

For more information about handling events, see .

*yyyyy) ToString*

#### *Description*

Occurs when when **System.Windows.Forms.AccessibleObject** is providing help to accessibility applications.

For more information about handling events, see .

*zzzzz) ToString*

[C#] public new event QueryContinueDragEventHandler QueryContinueDrag;

[C++] public: \_\_event QueryContinueDragEventHandler\* QueryContinueDrag;

[VB] Shadows Public Event QueryContinueDrag As

QueryContinueDragEventHandler

#### *Description*

Occurs during a drag-and-drop operation and allows the drag source to determine whether the drag-and-drop operation should be canceled.

For more information about handling events, see .



aaaaaa)ToString

Description

Occurs when the **System.Windows.Forms.Control.RightToLeft** property value has changed.

For more information about handling events, see .

bbbbbb)ToString

Description

Occurs when the control style has changed.

The **System.Windows.Forms.Control.StyleChanged** event occurs when **System.Windows.Forms.ControlStyles** flags ave been added or changed.

cccccc)ToString

Description

Occurs when the **System.Windows.Forms.Control.TabIndex** property value has changed.

For more information about handling events, see .

dddddd)ToString

[C#]      public      new      event      EventHandler      TabStopChanged;

[C++]      public:      \_\_event      EventHandler\*      TabStopChanged;

[VB]      Shadows      Public      Event      TabStopChanged      As      EventHandler

## Description

Occurs when the **System.Windows.Forms.Control.TabStop** property value has changed.

For more information about handling events, see .

*eeeeee)ToString*

[C#]	public	new	event	EventHandler	TextChanged;
[C++]	public:		__event	EventHandler*	TextChanged;
[VB]	Shadows	Public	Event	TextChanged	As EventHandler

## Description

Occurs when the **System.Windows.Forms.Control.Text** property value has changed.

For more information about handling events, see .

*ffffff) AttachInterfaces*

[C#]	protected	virtual	void	AttachInterfaces();
[C++]	protected:	virtual	void	AttachInterfaces();
[VB]	Overridable	Protected	Sub	AttachInterfaces()
[JScript]	protected	function		AttachInterfaces();

## Description

Called when the **System.Windows.Forms.AxHost** object is ready to create the underlying ActiveX object.

Classes that extend **System.Windows.Forms.AxHost** should override this method. Within an overridden version of this method, the extending class should

call **System.Windows.Forms.AxHost.GetOcx** to retrieve its own interface. In most cases, the **System.Windows.Forms.AxHost.GetOcx** method should not be called before this method is called.

### *gggggg)BeginInit*

[C#]	public		void	BeginInit();
[C++]	public:	__sealed	void	BeginInit();
[VB]	NotOverridable	Public	Sub	BeginInit()
[JScript]	public		function	BeginInit();

### *Description*

Begins the initialization of an **System.Windows.Forms.AxHost** object.

Design environments typically use this method to start the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.AxHost.EndInit** method ends the initialization. Using the **System.Windows.Forms.AxHost.BeginInit** and **System.Windows.Forms.AxHost.EndInit** methods prevents the control from being used before it is fully initialized. The initialization occurs at run time.

### *hhhhh)CreateHandle*

[C#]	protected	override	void	CreateHandle();
[C++]	protected:		void	CreateHandle();
[VB]	Overrides	Protected	Sub	CreateHandle()
[JScript]	protected	override	function	CreateHandle();

Creates a handle for this control. This method is called by the .NET framework, this should not be called.

### *iiii) CreateSink*

[C#]	protected	virtual	void	CreateSink();
------	-----------	---------	------	---------------

1 [C++] protected: virtual void CreateSink();

2 [VB] Overridable Protected Sub CreateSink()

3 [JScript] protected function CreateSink();

4 *jjjjj) DestroyHandle*

6 [C#] protected override void DestroyHandle();

7 [C++] protected: void DestroyHandle();

8 [VB] Overrides Protected Sub DestroyHandle()

9 [JScript] protected override function DestroyHandle(); Destroys the handle  
10 associated with this control. User code should in general not call this function.

11 *kkkkkk)DetachSink*

13 [C#] protected virtual void DetachSink();

14 [C++] protected: virtual void DetachSink();

15 [VB] Overridable Protected Sub DetachSink()

16 [JScript] protected function DetachSink();

17 *lllll) Dispose*

19 [C#] protected override void Dispose(bool disposing);

20 [C++] protected: void Dispose(bool disposing);

21 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

22 [JScript] protected override function Dispose(disposing : Boolean); Frees all  
23 resources associated with this control. This method may not be called at runtime.

Any resources used by the control should be setup to be released when the control is GC'ed. Inheriting classes should always call base.dispose.

### *mmmmm)DoVerb*

```
[C#]          public          void          DoVerb(int          verb);
[C++]          public:          void          DoVerb(int          verb);
[VB]    Public    Sub    DoVerb(ByVal    verb    As    Integer)
[JScript] public function DoVerb(verb : int);
```

### *nnnnnn)EndInit*

```
[C#]          public          void          EndInit();
[C++]          public:          __sealed          void          EndInit();
[VB]    NotOverridable          Public    Sub    EndInit()
[JScript]          public          function          EndInit();
```

### *Description*

Ends the initialization of an **System.Windows.Forms.AxHost** object.

Design environments typically use this method to start the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.AxHost.EndInit** method ends the initialization. Using the **System.Windows.Forms.AxHost.BeginInit** and **System.Windows.Forms.AxHost.EndInit** methods prevents the control from being used before it is fully initialized. The initialization occurs at run time.

### *ooooo)GetColorFromOleColor*

```
[C#]    protected    static    Color    GetColorFromOleColor(uint    color);
[C++]    protected:    static    Color    GetColorFromOleColor(unsigned int    color);
```

1 [VB] Protected Shared Function GetColorFromOleColor(ByVal color As UInt32)

2 As Color

3 [JScript] protected static function GetColorFromOleColor(color : UInt32) : Color;

4 Maps from an OLE COLOR to a System.Drawing.Color

5 *pppppp)GetFontFromIFont*

7 [C#] protected static Font GetFontFromIFont(object font);

8 [C++] protected: static Font\* GetFontFromIFont(Object\* font);

9 [VB] Protected Shared Function GetFontFromIFont(ByVal font As Object) As

10 Font

11 [JScript] protected static function GetFontFromIFont(font : Object) : Font; Maps

12 from an OLE IFont to a System.Drawing.Font object

13 *qqqqqq)GetFontFromIFontDisp*

15 [C#] protected static Font GetFontFromIFontDisp(object font);

16 [C++] protected: static Font\* GetFontFromIFontDisp(Object\* font);

17 [VB] Protected Shared Function GetFontFromIFontDisp(ByVal font As Object)

18 As Font

19 [JScript] protected static function GetFontFromIFontDisp(font : Object) : Font;

20 Maps from an IFontDisp to a System.Drawing.Font object

21 *rrrrrr) GetIFontDispFromFont*

23 [C#] protected static object GetIFontDispFromFont(Font font);

24 [C++] protected: static Object\* GetIFontDispFromFont(Font\* font);

1 [VB] Protected Shared Function GetIFontDispFromFont(ByVal font As Font) As  
2 Object

3 [JScript] protected static function GetIFontDispFromFont(font : Font) : Object;  
4 Maps from a System.Drawing.Font object to an OLE IFontDisp

5 *sssss) GetIFontFromFont*

6  
7 [C#] protected static object GetIFontFromFont(Font font);

8 [C++] protected: static Object\* GetIFontFromFont(Font\* font);

9 [VB] Protected Shared Function GetIFontFromFont(ByVal font As Font) As  
10 Object

11 [JScript] protected static function GetIFontFromFont(font : Font) : Object; Maps  
12 from a System.Drawing.Font object to an OLE IFont

13 *ttttt) GetIPictureDispFromPicture*

14  
15 [C#] protected static object GetIPictureDispFromPicture(Image image);

16 [C++] protected: static Object\* GetIPictureDispFromPicture(Image\* image);

17 [VB] Protected Shared Function GetIPictureDispFromPicture(ByVal image As  
18 Image) As Object

19 [JScript] protected static function GetIPictureDispFromPicture(image : Image) :  
20 Object; Maps from a System.Drawing.Image to an OLE IPictureDisp

21 *uuuuuu)GetIPictureFromCursor*

22  
23 [C#] protected static object GetIPictureFromCursor(Cursor cursor);

24 [C++] protected: static Object\* GetIPictureFromCursor(Cursor\* cursor);

1 [VB] Protected Shared Function GetIPictureFromCursor(ByVal cursor As Cursor)

2 As Object

3 [JScript] protected static function GetIPictureFromCursor(cursor : Cursor) :

4 Object; Maps from a System.Drawing.Cursor to an OLE IPicture

5 *vvvvvv)GetIPictureFromPicture*

7 [C#] protected static object GetIPictureFromPicture(Image image);

8 [C++] protected: static Object\* GetIPictureFromPicture(Image\* image);

9 [VB] Protected Shared Function GetIPictureFromPicture(ByVal image As Image)

10 As Object

11 [JScript] protected static function GetIPictureFromPicture(image : Image) :

12 Object; Maps from a System.Drawing.Image to an OLE IPicture

13 *wwwwww)GetOAdDateFromTime*

15 [C#] protected static double GetOAdDateFromTime(DateTime time);

16 [C++] protected: static double GetOAdDateFromTime(DateTime time);

17 [VB] Protected Shared Function GetOAdDateFromTime(ByVal time As DateTime)

18 As Double

19 [JScript] protected static function GetOAdDateFromTime(time : DateTime) :

20 double; Maps from a DateTime object to an OLE DATE (expressed as a double)

21 *xxxxxx)GetOcx*

23 [C#] public object GetOcx();

24 [C++] public: Object\* GetOcx();



1 [VB] Public Function GetOcx() As Object

2 [JScript] public function GetOcx() : Object;

4 *Description*

5 Retrieves a pointer to the underlying ActiveX control.

6 *Return Value:* A pointer to the underlying ActiveX control.

7 *yyyyyy)GetOleColorFromColor*

8 [C#] protected static uint GetOleColorFromColor(Color color);

9 [C++] protected: static unsigned int GetOleColorFromColor(Color color);

10 [VB] Protected Shared Function GetOleColorFromColor(ByVal color As Color)

11 As UInt32

12 [JScript] protected static function GetOleColorFromColor(color : Color) : UInt32;

13 Maps from an System.Drawing.Color to an OLE COLOR

14 *zzzzzz) GetPictureFromIPicture*

15 [C#] protected static Image GetPictureFromIPicture(object picture);

16 [C++] protected: static Image\* GetPictureFromIPicture(Object\* picture);

17 [VB] Protected Shared Function GetPictureFromIPicture(ByVal picture As

18 Object) As Image

19 [JScript] protected static function GetPictureFromIPicture(picture : Object) :

20 Image; Maps from an OLE IPicture to a System.Drawing.Image

21 *aaaaaaa)GetPictureFromIPictureDisp*

22 [C#] protected static Image GetPictureFromIPictureDisp(object picture);

```

1 [C++] protected: static Image* GetPictureFromIPictureDisp(Object* picture);
2 [VB] Protected Shared Function GetPictureFromIPictureDisp(ByVal picture As
3 Object) As Image
4 [JScript] protected static function GetPictureFromIPictureDisp(picture : Object) :
5 Image; Maps from an OLE IPictureDisp to a System.Drawing.Image

```

#### *bbbbbbb)GetTimeFromOAdate*

```

8 [C#] protected static DateTime GetTimeFromOAdate(double date);
9 [C++] protected: static DateTime GetTimeFromOAdate(double date);
10 [VB] Protected Shared Function GetTimeFromOAdate(ByVal date As Double)
11 As DateTime
12 [JScript] protected static function GetTimeFromOAdate(date : double) :
13 DateTime; Maps from an OLE DATE (expressed as a double) to a DateTime
14 object

```

#### *ccccccc)HasPropertyPages*

```

17 [C#] public bool HasPropertyPages();
18 [C++] public: bool HasPropertyPages();
19 [VB] Public Function HasPropertyPages() As Boolean
20 [JScript] public function HasPropertyPages() : Boolean;

```

#### *Description*

Determines if the ActiveX control has a property page.

*Return Value:* **true** if the ActiveX control has a property page; otherwise, **false** .

*ddddddd)InvokeEditMode*

```
1
2
3 [C#]          public          void          InvokeEditMode();
4 [C++]         public:         void          InvokeEditMode();
5 [VB]          Public          Sub           InvokeEditMode()
6 [JScript] public function InvokeEditMode();
```

*eeeeeee)IsInputKey*

```
8
9 [C#]    protected    override    bool    IsInputKey(Keys    keyData);
10 [C++]   protected:   bool        IsInputKey(Keys    keyData);
11 [VB]    Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
12 Boolean
13 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;
```

*Description*

Determines whether the specified key is a regular input key or a special key that requires preprocessing.

**Return Value:** **true** if the specified key is a regular input key; otherwise, **false** .

Call this method during window-message preprocessing to determine whether the specified key is a regular input key that should be sent directly to the control or a special key (such as PAGE UP and PAGE DOWN) that should be preprocessed. In the latter case, send the key to the control only if it is not consumed by the preprocessing phase. One of the **System.Windows.Forms.Keys** values.

*ffffff) MakeDirty*

```
23 [C#]          public          void          MakeDirty();
24 [C++]         public:         void          MakeDirty();
```

1 [VB] Public Sub MakeDirty()

2 [JScript] public function MakeDirty();

3 *ggggggg)OnBackColorChanged*

5 [C#] protected override void OnBackColorChanged(EventArgs e);

6 [C++] protected: void OnBackColorChanged(EventArgs\* e);

7 [VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)

8 [JScript] protected override function OnBackColorChanged(e : EventArgs);

9 *hhhhhhh)OnFontChanged*

11 [C#] protected override void OnFontChanged(EventArgs e);

12 [C++] protected: void OnFontChanged(EventArgs\* e);

13 [VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

14 [JScript] protected override function OnFontChanged(e : EventArgs);

15 *iiiiiii) OnForeColorChanged*

17 [C#] protected override void OnForeColorChanged(EventArgs e);

18 [C++] protected: void OnForeColorChanged(EventArgs\* e);

19 [VB] Overrides Protected Sub OnForeColorChanged(ByVal e As EventArgs)

20 [JScript] protected override function OnForeColorChanged(e : EventArgs);

21 *jjjjjjj) OnHandleCreated*

23 [C#] protected override void OnHandleCreated(EventArgs e);

24 [C++] protected: void OnHandleCreated(EventArgs\* e);

[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)  
 [JScript] protected override function OnHandleCreated(e : EventArgs); Inheriting  
 classes should override this method to find out when the handle has been created.  
 Call base.OnHandleCreated first.

#### *kkkkkkk)OnLostFocus*

[C#] protected override void OnLostFocus(EventArgs e);  
 [C++] protected: void OnLostFocus(EventArgs\* e);  
 [VB] Overrides Protected Sub OnLostFocus(ByVal e As EventArgs)  
 [JScript] protected override function OnLostFocus(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.LostFocus** event.

#### *llllll) PreProcessMessage*

[C#] public override bool PreProcessMessage(ref Message msg);  
 [C++] public: bool PreProcessMessage(Message\* msg);  
 [VB] Overrides Public Function PreProcessMessage(ByRef msg As Message) As  
 Boolean

[JScript] public override function PreProcessMessage(msg : Message) : Boolean;  
 This method is called by the application's message loop to pre-process input  
 messages before they are dispatched. Possible values for the msg.message field are  
 WM\_KEYDOWN, WM\_SYSKEYDOWN, WM\_CHAR, and WM\_SYSCHAR.  
 If this method processes the message it must return true, in which case the  
 message loop will not dispatch the message. This method should not be called

1 directly by the user. The keyboard processing of input keys to AxHost controls go  
 2 in 3 steps inside AxHost.PreProcessMessage() (1) Call the OCX's  
 3 TranslateAccelerator. This may or may not call back into us using  
 4 IOleControlSite::TranslateAccelerator() (2) If the control completely processed  
 5 this without calling us back: -- If this returns S\_OK, then it means that the control  
 6 already processed this message and we return true, forcing us to not do any more  
 7 processing or dispatch the message. -- If this returns S\_FALSE, then it means that  
 8 the control wants us to dispatch the message without doing any processing on our  
 9 side. (3) If the control completely processed this by calling us back: -- If this  
 10 returns S\_OK, then it means that the control processed this message and we return  
 11 true, forcing us to not do any more processing or dispatch the message. -- If this  
 12 returns S\_FALSE, then it means that the control did not process this message, but  
 13 we did, and so we should route it through our PreProcessMessage().

#### 14 *mmmmmmmm)ProcessMnemonic*

15  
 16 [C#] protected override bool ProcessMnemonic(char charCode);  
 17 [C++] protected: bool ProcessMnemonic(\_\_wchar\_t charCode);  
 18 [VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)  
 19 As Boolean  
 20 [JScript] protected override function ProcessMnemonic(charCode : Char) :  
 21 Boolean; Process a mnemonic character. This is done by manufacturing a  
 22 WM\_SYSKEYDOWN message and passing it to the ActiveX control.

***nnnnnnn)PropsValid***

```
1
2
3 [C#]                protected                bool                PropsValid();
4 [C++]                protected:                bool                PropsValid();
5 [VB]    Protected    Function    PropsValid()    As    Boolean
6 [JScript] protected function PropsValid() : Boolean;
```

***oooooooo)RaiseOnMouseDown***

```
7
8
9 [C#] protected void RaiseOnMouseDown(short button, short shift, int x, int y);
10 [C++] protected: void RaiseOnMouseDown(short button, short shift, int x, int y);
11 [VB] Protected Sub RaiseOnMouseDown(ByVal button As Short, ByVal shift As
12 Short,    ByVal    x    As    Integer,    ByVal    y    As    Integer)
13 [JScript] protected function RaiseOnMouseDown(button : Int16, shift : Int16, x :
14 int, y : int);
```

***ppppppp)RaiseOnMouseDown***

```
15
16
17 [C#] protected void RaiseOnMouseDown(short button, short shift, float x, float y);
18 [C++] protected: void RaiseOnMouseDown(short button, short shift, float x, float
19 y);
20 [VB] Protected Sub RaiseOnMouseDown(ByVal button As Short, ByVal shift As
21 Short,    ByVal    x    As    Single,    ByVal    y    As    Single)
22 [JScript] protected function RaiseOnMouseDown(button : Int16, shift : Int16, x :
23 float, y : float);
24
25
```

**qqqqqqq)RaiseOnMouseDown**

[C#] protected void RaiseOnMouseDown(object o1, object o2, object o3, object o4);

[C++] protected: void RaiseOnMouseDown(Object\* o1, Object\* o2, Object\* o3, Object\* o4);

[VB] Protected Sub RaiseOnMouseDown(ByVal o1 As Object, ByVal o2 As Object, ByVal o3 As Object, ByVal o4 As Object)

[JScript] protected function RaiseOnMouseDown(o1 : Object, o2 : Object, o3 : Object, o4 : Object);

**rrrrrrr)RaiseOnMouseMove**

[C#] protected void RaiseOnMouseMove(short button, short shift, int x, int y);

[C++] protected: void RaiseOnMouseMove(short button, short shift, int x, int y);

[VB] Protected Sub RaiseOnMouseMove(ByVal button As Short, ByVal shift As Short, ByVal x As Integer, ByVal y As Integer)

[JScript] protected function RaiseOnMouseMove(button : Int16, shift : Int16, x : int, y : int);

**sssssss)RaiseOnMouseMove**

[C#] protected void RaiseOnMouseMove(short button, short shift, float x, float y);

[C++] protected: void RaiseOnMouseMove(short button, short shift, float x, float y);

[VB] Protected Sub RaiseOnMouseMove(ByVal button As Short, ByVal shift As Short, ByVal x As Single, ByVal y As Single)



1 [JScript] protected function RaiseOnMouseMove(button : Int16, shift : Int16, x :  
2 float, y : float);

3 *tttttt) RaiseOnMouseMove*

5 [C#] protected void RaiseOnMouseMove(object o1, object o2, object o3, object  
6 o4);

7 [C++] protected: void RaiseOnMouseMove(Object\* o1, Object\* o2, Object\* o3,  
8 Object\* o4);

9 [VB] Protected Sub RaiseOnMouseMove(ByVal o1 As Object, ByVal o2 As  
10 Object, ByVal o3 As Object, ByVal o4 As Object)

11 [JScript] protected function RaiseOnMouseMove(o1 : Object, o2 : Object, o3 :  
12 Object, o4 : Object);

13 *uuuuuuu)RaiseOnMouseUp*

15 [C#] protected void RaiseOnMouseUp(short button, short shift, int x, int y);

16 [C++] protected: void RaiseOnMouseUp(short button, short shift, int x, int y);

17 [VB] Protected Sub RaiseOnMouseUp(ByVal button As Short, ByVal shift As  
18 Short, ByVal x As Integer, ByVal y As Integer)

19 [JScript] protected function RaiseOnMouseUp(button : Int16, shift : Int16, x : int,  
20 y : int);

21 *vvvvvvv)RaiseOnMouseUp*

23 [C#] protected void RaiseOnMouseUp(short button, short shift, float x, float y);

24 [C++] protected: void RaiseOnMouseUp(short button, short shift, float x, float y);

```

1 [VB] Protected Sub RaiseOnMouseUp(ByVal button As Short, ByVal shift As
2 Short, ByVal x As Single, ByVal y As Single)
3 [JScript] protected function RaiseOnMouseUp(button : Int16, shift : Int16, x :
4 float, y : float);

```

*wwwwww)RaiseOnMouseUp*

```

7 [C#] protected void RaiseOnMouseUp(object o1, object o2, object o3, object o4);

```

```

8 [C++] protected: void RaiseOnMouseUp(Object* o1, Object* o2, Object* o3,
9 Object* o4);

```

```

10 [VB] Protected Sub RaiseOnMouseUp(ByVal o1 As Object, ByVal o2 As Object,
11 ByVal o3 As Object, ByVal o4 As Object)

```

```

12 [JScript] protected function RaiseOnMouseUp(o1 : Object, o2 : Object, o3 :
13 Object, o4 : Object);

```

*xxxxxxx)SetAboutBoxDelegate*

```

16 [C#] protected void SetAboutBoxDelegate(AxHost.AboutBoxDelegate d);

```

```

17 [C++] protected: void SetAboutBoxDelegate(AxHost.AboutBoxDelegate* d);

```

```

18 [VB] Protected Sub SetAboutBoxDelegate(ByVal d As
19 AxHost.AboutBoxDelegate)

```

```

20 [JScript] protected function SetAboutBoxDelegate(d :
21 AxHost.AboutBoxDelegate);

```

### *Description*

Calls the **System.Windows.Forms.AxHost.ShowAboutBox** method to display the ActiveX controls about dialog box.

The

**System.Windows.Forms.AxHost.SetAboutBoxDelegate(System.Windows.Forms.AxHost.AboutBoxDelegate)** method also allows derived classes to handle the **System.Windows.Forms.AxHost.ShowAboutBox** method without attaching a delegate. This is the preferred technique for handling the **System.Windows.Forms.AxHost.ShowAboutBox** method in a derived class. The delegate to call.

*yyyyyyy)SetBoundsCore*

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified); Performs the work of setting the bounds of this control. User code should usually not call this function.

*zzzzzzz)SetVisibleCore*

[C#] protected override void SetVisibleCore(bool value);

[C++] protected: void SetVisibleCore(bool value);

[VB] Overrides Protected Sub SetVisibleCore(ByVal value As Boolean)

[JScript] protected override function SetVisibleCore(value : Boolean);

*aaaaaaa)ShowAboutBox*

[C#]	public	void	ShowAboutBox();
[C++]	public:	void	ShowAboutBox();
[VB]	Public	Sub	ShowAboutBox()
[JScript]	public	function	ShowAboutBox();

*Description*

Displays the ActiveX control's about dialog box.

If **System.Windows.Forms.AxHost.HasAboutBox** is **false** , no about dialog box is displayed.

*bbbbbbbb)ShowPropertyPages*

[C#]	public	void	ShowPropertyPages();
[C++]	public:	void	ShowPropertyPages();
[VB]	Public	Sub	ShowPropertyPages()

[JScript] public function ShowPropertyPages(); Displays the property page associated with the ActiveX control.

*Description*

Displays the property page associated with the ActiveX control.

If **System.Windows.Forms.AxHost.HasPropertyPages** returns **false** , no property pages are displayed.

*ccccccc)ShowPropertyPages*

[C#]	public	void	ShowPropertyPages(Control	control);
------	--------	------	---------------------------	-----------

1 [C++] public: void ShowPropertyPages(Control\* control);

2 [VB] Public Sub ShowPropertyPages(ByVal control As Control)

3 [JScript] public function ShowPropertyPages(control : Control);

4  
5 *Description*

6 Displays the property page associated with the ActiveX control assigned to the specified parent control.

7 If **System.Windows.Forms.AxHost.HasPropertyPages** returns **false** , no  
8 property pages are displayed. The parent **System.Windows.Forms.Control** of  
9 the ActiveX control.

10 *ddddddd)ICustomTypeDescriptor.GetAttributes*

11 [C#] AttributeCollection ICustomTypeDescriptor.GetAttributes();

12 [C++] AttributeCollection\* ICustomTypeDescriptor::GetAttributes();

13 [VB] Function GetAttributes() As AttributeCollection Implements  
14 ICustomTypeDescriptor.GetAttributes

15 [JScript] function ICustomTypeDescriptor.GetAttributes() : AttributeCollection;

16  
17 *eeeeeeee)ICustomTypeDescriptor.GetClassName*

18  
19 [C#] string ICustomTypeDescriptor.GetClassName();

20 [C++] String\* ICustomTypeDescriptor::GetClassName();

21 [VB] Function GetClassName() As String Implements  
22 ICustomTypeDescriptor.GetClassName

23 [JScript] function ICustomTypeDescriptor.GetClassName() : String;

***ffffff)ICustomTypeDescriptor.GetComponentName***

```
1
2
3 [C#]          string          ICustomTypeDescriptor.GetComponentName();
4 [C++]         String*         ICustomTypeDescriptor::GetComponentName();
5 [VB]   Function  GetComponentName()  As   String   Implements
6 ICustomTypeDescriptor.GetComponentName
7 [JScript] function ICustomTypeDescriptor.GetComponentName() : String;
```

***gggggggg)ICustomTypeDescriptor.GetConverter***

```
9
10 [C#]          TypeConverter      ICustomTypeDescriptor.GetConverter();
11 [C++]         TypeConverter*     ICustomTypeDescriptor::GetConverter();
12 [VB]   Function  GetConverter()   As   TypeConverter   Implements
13 ICustomTypeDescriptor.GetConverter
14 [JScript] function ICustomTypeDescriptor.GetConverter() : TypeConverter;
```

***hhhhhhh)ICustomTypeDescriptor.GetDefaultEvent***

```
16
17 [C#]          EventDescriptor     ICustomTypeDescriptor.GetDefaultEvent();
18 [C++]         EventDescriptor*    ICustomTypeDescriptor::GetDefaultEvent();
19 [VB]   Function  GetDefaultEvent() As   EventDescriptor   Implements
20 ICustomTypeDescriptor.GetDefaultEvent
21 [JScript] function ICustomTypeDescriptor.GetDefaultEvent() : EventDescriptor;
```

***iiiiiii) ICustomTypeDescriptor.GetDefaultProperty***

```
23
24 [C#]          PropertyDescriptor  ICustomTypeDescriptor.GetDefaultProperty();
25 [C++]         PropertyDescriptor* ICustomTypeDescriptor::GetDefaultProperty();
```

```

1  [VB] Function GetDefaultProperty() As PropertyDescriptor Implements
2  ICustomTypeDescriptor.GetDefaultProperty
3  [JScript] function ICustomTypeDescriptor.GetDefaultProperty() :
4  PropertyDescriptor;
5      jjjjjjj) ICustomTypeDescriptor.GetEditor
6
7  [C#] object ICustomTypeDescriptor.GetEditor(Type editorBaseType);
8  [C++] Object* ICustomTypeDescriptor::GetEditor(Type* editorBaseType);
9  [VB] Function GetEditor(ByVal editorBaseType As Type) As Object Implements
10 ICustomTypeDescriptor.GetEditor
11 [JScript] function ICustomTypeDescriptor.GetEditor(editorBaseType : Type) :
12 Object;
13      kkkkkkkk) ICustomTypeDescriptor.GetEvents
14
15 [C#] EventDescriptorCollection ICustomTypeDescriptor.GetEvents();
16 [C++] EventDescriptorCollection* ICustomTypeDescriptor::GetEvents();
17 [VB] Function GetEvents() As EventDescriptorCollection Implements
18 ICustomTypeDescriptor.GetEvents
19 [JScript] function ICustomTypeDescriptor.GetEvents() :
20 EventDescriptorCollection;
21      lllllll) ICustomTypeDescriptor.GetEvents
22
23 [C#] EventDescriptorCollection ICustomTypeDescriptor.GetEvents(Attribute[]
24 attributes);
25

```

```

1 [C++] EventDescriptorCollection* ICustomTypeDescriptor::GetEvents(Attribute*
2 attributes[]);
3 [VB] Function GetEvents(ByVal attributes() As Attribute) As
4 EventDescriptorCollection Implements ICustomTypeDescriptor.GetEvents
5 [JScript] function ICustomTypeDescriptor.GetEvents(attributes : Attribute[]) :
6 EventDescriptorCollection;
7
8 mmmmmmmmmm)ICustomTypeDescriptor.GetProperties
9
10 [C#] PropertyDescriptorCollection ICustomTypeDescriptor.GetProperties();
11 [C++] PropertyDescriptorCollection* ICustomTypeDescriptor::GetProperties();
12 [VB] Function GetProperties() As PropertyDescriptorCollection Implements
13 ICustomTypeDescriptor.GetProperties
14 [JScript] function ICustomTypeDescriptor.GetProperties() :
15 PropertyDescriptorCollection;
16
17 nnnnnnnnn)ICustomTypeDescriptor.GetProperties
18
19 [C#] PropertyDescriptorCollection
20 ICustomTypeDescriptor.GetProperties(Attribute[] attributes);
21 [C++] PropertyDescriptorCollection*
22 ICustomTypeDescriptor::GetProperties(Attribute* attributes[]);
23 [VB] Function GetProperties(ByVal attributes() As Attribute) As
24 PropertyDescriptorCollection Implements ICustomTypeDescriptor.GetProperties
25 [JScript] function ICustomTypeDescriptor.GetProperties(attributes : Attribute[]) :
26 PropertyDescriptorCollection;

```



*oooooooo)ICustomPropertyDescriptor.GetPropertyOwner*

```
1
2
3 [C#] object ICustomPropertyDescriptor.GetPropertyOwner(PropertyDescriptor pd);
4 [C++] Object* ICustomPropertyDescriptor::GetPropertyOwner(PropertyDescriptor*
5 pd);
6 [VB] Function GetPropertyOwner(ByVal pd As PropertyDescriptor) As Object
7 Implements ICustomPropertyDescriptor.GetPropertyOwner
8 [JScript] function ICustomPropertyDescriptor.GetPropertyOwner(pd :
9 PropertyDescriptor) : Object;
```

*pppppppp)WndProc*

```
10
11
12 [C#] protected override void WndProc(ref Message m);
13 [C++] protected: void WndProc(Message* m);
14 [VB] Overrides Protected Sub WndProc(ByRef m As Message)
15 [JScript] protected override function WndProc(m : Message); AxHost wndProc.
```

16 All messages are sent to wndProc after getting filtered through the  
17 preProcessMessage function. Certain messages are forwarder directly to the  
18 ActiveX control, others are first processed by the wndProc of Control

19 BaseCollection class (System.Windows.Forms)

20 *a) WndProc*

21  
22  
23 *Description*

24 Provides the base functionality for creating data-related collections in the  
25 **System.Windows.Forms** namespace.

The **System.Windows.Forms.BaseCollection** class is not intended for use by application developers. Application developers should use **System.Collections.CollectionBase** instead.

*b) BaseCollection*

*Example Syntax:*

*c) WndProc*

[C#] public BaseCollection();

[C++] public: BaseCollection();

[VB] Public Sub New()

[JScript] public function BaseCollection();

*d) Count*

*e) WndProc*

[C#] public virtual int Count {get;}

[C++] public: \_\_property virtual int get\_Count();

[VB] Overridable Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

*Description*

Gets the total number of elements in the collection.

*f) IsReadOnly*

*g) WndProc*

[C#] public bool IsReadOnly {get;}

```

1  [C++]      public:      __property      bool      get_IsReadOnly();
2  [VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
3  [JScript]  public      function      get      IsReadOnly()      :      Boolean;

```

#### *Description*

Gets a value indicating whether the collection is read-only.

The collections which derive from the **System.Windows.Forms.BaseCollection** class are always writeable, which is why this property always returns **false** .

*h) IsSynchronized*

*i) WndProc*

```

12 [C#]      public      bool      IsSynchronized      {get;}
13 [C++]      public:      __property      bool      get_IsSynchronized();
14 [VB]      Public      ReadOnly      Property      IsSynchronized      As      Boolean
15 [JScript]  public      function      get      IsSynchronized()      :      Boolean;

```

#### *Description*

Gets a value indicating whether access to the **System.Collections.ICollection** is synchronized.

If a collection is thread safe, the **System.Windows.Forms.BaseCollection.IsSynchronized** property returns **true** , and the programmer does not have to do anything to keep the collection safe.

j) *List*

k) *WndProc*

```
[C#]      protected      virtual      ArrayList      List      {get;}
[C++]     protected:     __property  virtual      ArrayList*  get_List();
[VB]      Overridable    Protected    ReadOnly    Property    List    As    ArrayList
[JScript] protected      function    get      List()      :      ArrayList;
```

#### *Description*

Gets the list of elements contained in the  
**System.Windows.Forms.BaseCollection** instance.

l) *SyncRoot*

m) *WndProc*

```
[C#]      public          object          SyncRoot          {get;}
[C++]     public:         __property      Object*           get_SyncRoot();
[VB]      Public          ReadOnly        Property          SyncRoot      As      Object
[JScript] public          function        get      SyncRoot()      :      Object;
```

#### *Description*

Gets an object that can be used to synchronize access to the  
**System.Windows.Forms.BaseCollection** .

If, as is the case with the **System.Windows.Forms.BaseCollection** , the  
**System.Windows.Forms.BaseCollection.IsSynchronized** property returns  
**false** , then the **System.Windows.Forms.BaseCollection.SyncRoot**  
property returns an object that can be used with the C# **lock** keyword.

### n) *CopyTo*

```
[C#]      public      void      CopyTo(Array      ar,      int      index);  
[C++]     public:     __sealed   void      CopyTo(Array*   ar,      int      index);  
[VB]      NotOverridable Public Sub CopyTo(ByVal ar As Array, ByVal index As  
Integer)  
[JScript] public      function CopyTo(ar      :      Array,      index      :      int);
```

#### *Description*

Copies all the elements of the current one-dimensional **System.Array** to the specified one-dimensional **System.Array** starting at the specified destination **System.Array** index. The one-dimensional **System.Array** that is the destination of the elements copied from the current Array. The zero-based relative index in *ar* at which copying begins.

### o) *GetEnumerator*

```
[C#]      public      IEnumerator      GetEnumerator();  
[C++]     public:     __sealed   IEnumerator*      GetEnumerator();  
[VB]      NotOverridable Public Function GetEnumerator() As IEnumerator  
[JScript] public      function      GetEnumerator()      :      IEnumerator;
```

#### *Description*

Gets the object that allows iterating through the members of the collection.  
*Return Value:* An object that implements the **System.Collections.IEnumerator** interface.

Binding class (System.Windows.Forms)

*a) ToString*

*Description*

Represents the simple binding between the property value of an object and the property value of a control.

Use the **System.Windows.Forms.Binding** class to create and maintain a simple binding between the property of a control and either the property of an object, or the property of the current object in a list of objects.

*b) Binding*

*Example Syntax:*

*c) ToString*

[C#] public Binding(string propertyName, object dataSource, string dataMember);

[C++] public: Binding(String\* propertyName, Object\* dataSource, String\* dataMember);

[VB] Public Sub New(ByVal propertyName As String, ByVal dataSource As Object, ByVal dataMember As String)

[JScript] public function Binding(propertyName : String, dataSource : Object, dataMember : String); Initializes a new instance of the **System.Windows.Forms.Binding** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.Binding** class that simple-binds the specified control property to the specified data member of the specified data source.

You can specify an instance of any of the following classes for the data source:  
**System.Data.DataSetSystem.Data.DataTableSystem.Data.DataViewSystem.Data.DataViewManager** Any class that implements the **System.Collections.IList** interface. The name of the control property to bind.  
An **System.Object** that represents the data source. The property or list to bind to.

d) *BindingManagerBase*

e) *ToString*

[C#] public BindingManagerBase BindingManagerBase {get;}

[C++] public: \_\_property BindingManagerBase\* get\_BindingManagerBase();

[VB] Public ReadOnly Property BindingManagerBase As BindingManagerBase

[JScript] public function get BindingManagerBase() : BindingManagerBase;

#### *Description*

Gets this binding's **System.Windows.Forms.BindingManagerBase**.

Use the **System.Windows.Forms.BindingManagerBase** to iterate through a data-bound list by incrementing or decrementing the **System.Windows.Forms.BindingManagerBase.Position** property. The **System.Windows.Forms.BindingManagerBase** class is abstract. The **System.Windows.Forms.CurrencyManager** class, which manages data-bound lists, inherits from the **System.Windows.Forms.BindingManagerBase** class.

f) *BindingMemberInfo*

g) *ToString*

[C#] public BindingMemberInfo BindingMemberInfo {get;}

[C++] public: \_\_property BindingMemberInfo get\_BindingMemberInfo();

[VB] Public ReadOnly Property BindingMemberInfo As BindingMemberInfo

[JScript] public function get BindingMemberInfo() : BindingMemberInfo;

## Description

Gets an object that contains information about this binding based on the *dataMember* parameter in the **System.Windows.Forms.Binding.#ctor** constructor.

The **System.Windows.Forms.BindingMemberInfo** is created from the *dataMember* string passed to the **System.Windows.Forms.Binding.#ctor** constructor.

h) *Control*

i) *ToString*

[C#]	public	Control	Control	{get;}
[C++]	public:	__property	Control*	get_Control();
[VB]	Public	ReadOnly	Property	Control As Control
[JScript]	public	function	get	Control() : Control;

## Description

Gets the control that the binding belongs to.

j) *DataSource*

k) *ToString*

[C#]	public	object	DataSource	{get;}
[C++]	public:	__property	Object*	get_DataSource();
[VB]	Public	ReadOnly	Property	DataSource As Object
[JScript]	public	function	get	DataSource() : Object;



## Description

Gets the data source for this binding.

Possible data sources include:

**System.Data.DataSetSystem.Data.DataTableSystem.Data.DataViewSystem.Data.DataViewManager** Any object that implements the **System.Collections.IList** interface.

l) *IsBinding*

m) *ToString*

[C#]	public	bool	IsBinding	{get;}
[C++]	public:	__property	bool	get_IsBinding();
[VB]	Public	ReadOnly	Property	IsBinding As Boolean
[JScript]	public	function	get	IsBinding() : Boolean;

## Description

Gets a value indicating whether the binding is active.

A binding is active when it meets three conditions: all of its properties are set, it belongs to a **System.Windows.Forms.BindingsCollection** that points to a table of a data source, and the data source is not **null**.

n) *PropertyName*

o) *ToString*

[C#]	public	string	PropertyName	{get;}
[C++]	public:	__property	String*	get_PropertyName();
[VB]	Public	ReadOnly	Property	PropertyName As String
[JScript]	public	function	get	PropertyName() : String;

## Description

Gets or sets the name of the control's data-bound property.

Use the **System.Windows.Forms.Binding.PropertyName** to specify the control property that you want to bind to a list in a data source. Most commonly, you bind a display property such as the

**System.Windows.Forms.Control.Text** property of a

**System.Windows.Forms.TextBox** control. However, because you can bind any property of a control, you can programmatically create controls at run time using data from a database.

### p) ToString

[C#]	public	event	ConvertEventHandler	Format;
[C++]	public:	__event	ConvertEventHandler*	Format;
[VB]	Public	Event	Format As	ConvertEventHandler

## Description

Occurs when the property of a control is bound to a data value.

The **System.Windows.Forms.Binding.Format** and **System.Windows.Forms.Binding.Parse** events allow you to create custom formats for displaying data. For example, if the data in a table is of type **System.Decimal**, you can display the data in the local currency format by setting the **System.Windows.Forms.ConvertEventArgs.Value** property of the **System.Windows.Forms.ConvertEventArgs** object to the formatted value in the **System.Windows.Forms.Binding.Format** event. You must consequently unformat the displayed value in the **System.Windows.Forms.Binding.Parse** event.

### q) ToString

[C#]	public	event	ConvertEventHandler	Parse;
[C++]	public:	__event	ConvertEventHandler*	Parse;

[VB]        Public        Event        Parse        As        ConvertEventHandler

### *Description*

Occurs when the value of a data-bound control changes.

The **System.Windows.Forms.Binding.Format** and **System.Windows.Forms.Binding.Parse** events allow you to create custom formats for displaying data. For example, if the data in a table is of type **System.Decimal**, you can display the data in the local currency format by setting the **System.Windows.Forms.ConvertEventArgs.Value** property of the **System.Windows.Forms.ConvertEventArgs** object to the formatted value in the **System.Windows.Forms.Binding.Format** event. You must consequently unformat the displayed value in the **System.Windows.Forms.Binding.Parse** event.

#### *r)        OnFormat*

[C#]        protected        virtual        void        OnFormat(ConvertEventArgs        cevent);

[C++]        protected:        virtual        void        OnFormat(ConvertEventArgs\*        cevent);

[VB]        Overridable Protected Sub OnFormat(ByVal cevent As ConvertEventArgs)

[JScript]        protected        function        OnFormat(cevent        :        ConvertEventArgs);

### *Description*

Raises the **System.Windows.Forms.Binding.Format** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.ConvertEventArgs** that contains the event data.

#### *s)        OnParse*

[C#]        protected        virtual        void        OnParse(ConvertEventArgs        cevent);

[C++]        protected:        virtual        void        OnParse(ConvertEventArgs\*        cevent);

[VB] Overridable Protected Sub OnParse(ByVal cevent As ConvertEventArgs)  
[JScript] protected function OnParse(cevent : ConvertEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Binding.Parse** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.ConvertEventArgs** that contains the event data.

BindingContext class (System.Windows.Forms)

#### *a) ToString*

#### *Description*

Manages the collection of **System.Windows.Forms.BindingManagerBase** objects for any object that inherits from the **System.Windows.Forms.Control** class.

Each object that inherits from the **System.Windows.Forms.Control** class can have a single **System.Windows.Forms.BindingContext** object. That **System.Windows.Forms.BindingContext** manages the **System.Windows.Forms.BindingManagerBase** objects for that control and any contained controls. Use the **System.Windows.Forms.BindingContext** to create or return the **System.Windows.Forms.BindingManagerBase** for a data source used by the contained data-bound controls. Most commonly, you use the **System.Windows.Forms.Form** class's **System.Windows.Forms.BindingContext** to return **System.Windows.Forms.BindingManagerBase** objects for the data-bound controls on the form. If you use a container control, such as a **System.Windows.Forms.GroupBox** , **System.Windows.Forms.Panel** , or **System.Windows.Forms.TabControl** , to contain data-bound controls, you can create a **System.Windows.Forms.BindingContext** for just that container control and its controls. This allows each part of your form to be managed by its own **System.Windows.Forms.BindingManagerBase** object. See the **System.Windows.Forms.BindingContext.#ctor** constructor for more details on creating multiple **System.Windows.Forms.BindingManagerBase** objects for the same data source.

b) *BindingContext*

*Example Syntax:*

c) *ToString*

[C#]	public	BindingContext();
[C++]	public:	BindingContext();
[VB]	Public	Sub New()
[JScript]	public	function BindingContext();

*Description*

Initializes a new instance of the **System.Windows.Forms.BindingContext** class.

Create a new **System.Windows.Forms.BindingContext** and set it to the **System.Windows.Forms.BindingContext** property of an object that inherits from **System.Windows.Forms.Control** when you want to have multiple **System.Windows.Forms.BindingManagerBase** instances for the same data source.

d) *IsReadOnly*

e) *ToString*

[C#]	public	bool	IsReadOnly	{get;}
[C++]	public:	__property	bool	get_IsReadOnly();
[VB]	Public	ReadOnly	Property	IsReadOnly As Boolean
[JScript]	public	function	get	IsReadOnly() : Boolean;

*Description*

Gets a value indicating whether the collection is read-only.

The property is derived from **System.Collections.ICollection** , and is overridden to always return false.

*f) Item*

*g) ToString*

```
[C#] public BindingManagerBase this[object dataSource] {get;}
```

```
[C++] public: __property BindingManagerBase* get_Item(Object* dataSource);
```

```
[VB] Public Default ReadOnly Property Item(ByVal dataSource As Object) As  
BindingManagerBase
```

```
[JScript] returnValue = BindingContextObject.Item(dataSource); Gets a particular  
System.Windows.Forms.BindingManagerBase
```

### *Description*

Gets the **System.Windows.Forms.BindingManagerBase** that is associated with the specified data source.

Use this constructor if the **System.Windows.Forms.BindingManagerBase** you want does not require a navigation path. For example, if the **System.Windows.Forms.BindingManagerBase** manages a set of **System.Windows.Forms.Binding** objects that use an **System.Collections.ArrayList** or **System.Data.DataTable** as the **System.Windows.Forms.Binding.DataSource** , no navigation path is required. The data source associated with a particular **System.Windows.Forms.BindingManagerBase**.

*h) Item*

*i) ToString*

```
[C#] public BindingManagerBase this[object dataSource, string dataMember]  
{get;}
```

```
[C++] public: __property BindingManagerBase* get_Item(Object* dataSource,
```

```

1 String* dataMember);
2 [VB] Public Default ReadOnly Property Item(ByVal dataSource As Object, ByVal
3 dataMember As String) As BindingManagerBase
4 [JScript] returnValue = BindingContextObject.Item(dataSource, dataMember);
5

```

### *Description*

Gets the **System.Windows.Forms.BindingManagerBase** that is associated with the specified data source and data member.

Use this constructor when the **System.Windows.Forms.BindingManagerBase** manages a set of **System.Windows.Forms.Binding** objects for which the data source contains multiple objects. For example, a **System.Data.DataSet** can contain several **System.Data.DataTable** objects linked by **System.Data.DataRelation** objects. In such a case, the navigation path is required to enable the **System.Windows.Forms.BindingContext** to return the correct **System.Windows.Forms.BindingManagerBase**. The data source associated with a particular **System.Windows.Forms.BindingManagerBase**. A navigation path containing the information that resolves to a specific **System.Windows.Forms.BindingManagerBase**.

### *j) ToString*

```

17 [C#] public event CollectionChangeEventHandler CollectionChanged;
18 [C++] public: __event CollectionChangeEventHandler* CollectionChanged;
19 [VB] Public Event CollectionChanged As CollectionChangeEventHandler
20

```

### *Description*

Occurs when the collection has changed.

For more information about handling events, see .

*k) Add*

[C#] protected internal void Add(object dataSource, BindingManagerBase listManager);

[C++] protected public: void Add(Object\* dataSource, BindingManagerBase\* listManager);

[VB] Protected Friend Dim Sub Add(ByVal dataSource As Object, ByVal listManager As BindingManagerBase)

[JScript] package function Add(dataSource : Object, listManager : BindingManagerBase);

*Description*

Adds the **System.Windows.Forms.BindingManagerBase** associated with a specific data source to the collection. The **System.Object** associated with the **System.Windows.Forms.BindingManagerBase**. The **System.Windows.Forms.BindingManagerBase** to add.

*l) AddCore*

[C#] protected virtual void AddCore(object dataSource, BindingManagerBase listManager);

[C++] protected: virtual void AddCore(Object\* dataSource, BindingManagerBase\* listManager);

[VB] Overridable Protected Sub AddCore(ByVal dataSource As Object, ByVal listManager As BindingManagerBase)

[JScript] protected function AddCore(dataSource : Object, listManager : BindingManagerBase);



*Description*

*m) Clear*

[C#]	protected	internal	void	Clear();
[C++]	protected	public:	void	Clear();
[VB]	Protected	Friend	Dim Sub	Clear()
[JScript]	package	function		Clear();

*Description*

Clears the collection of any **System.Windows.Forms.BindingManagerBase** objects.

*n) ClearCore*

[C#]	protected	virtual	void	ClearCore();
[C++]	protected:	virtual	void	ClearCore();
[VB]	Overridable	Protected	Sub	ClearCore()
[JScript]	protected	function		ClearCore();

*Description*

Clears the collection.

*o) Contains*

[C#]	public	bool	Contains(object	dataSource);
[C++]	public:	bool	Contains(Object*	dataSource);

[VB] Public Function Contains(ByVal dataSource As Object) As Boolean  
[JScript] public function Contains(dataSource : Object) : Boolean; Gets a value indicating whether the **System.Windows.Forms.BindingContext** contains the specified **System.Windows.Forms.BindingManagerBase**.

#### *Description*

Gets a value indicating whether the **System.Windows.Forms.BindingContext** contains the **System.Windows.Forms.BindingManagerBase** associated with the specified data source.

*Return Value:* **true**, if the **System.Windows.Forms.BindingContext** contains the specified **System.Windows.Forms.BindingManagerBase**; otherwise, **false**.

See the **System.Windows.Forms.Binding** class for a list of possible data sources and details on creating bindings between controls and data sources. An **System.Object** that represents the data source.

#### *p) Contains*

[C#] public bool Contains(object dataSource, string dataMember);  
[C++] public: bool Contains(Object\* dataSource, String\* dataMember);  
[VB] Public Function Contains(ByVal dataSource As Object, ByVal dataMember As String) As Boolean  
[JScript] public function Contains(dataSource : Object, dataMember : String) : Boolean;

#### *Description*

Gets a value indicating whether the **System.Windows.Forms.BindingContext** contains the **System.Windows.Forms.BindingManagerBase** associated with the specified data source and data member.

*Return Value:* **true** if the **System.Windows.Forms.BindingContext** contains the specified **System.Windows.Forms.BindingManagerBase** ; otherwise, **false** .

See the **System.Windows.Forms.Binding** class for a list of possible data sources and for details on creating bindings between controls and data sources. An **System.Object** that represents the data source. The information needed to resolve to a specific **System.Windows.Forms.BindingManagerBase**.

*q) OnCollectionChanged*

[C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs ccevent);

[C++] protected: virtual void OnCollectionChanged(CollectionChangeEventArgs\* ccevent);

[VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As CollectionChangeEventArgs)

[JScript] protected function OnCollectionChanged(ccevent : CollectionChangeEventArgs);

*Description*

This method is called whenever the collection changes. Overriders of this method should call the base implementation of this method. The event information.

*r) Remove*

[C#] protected internal void Remove(object dataSource);

[C++] protected public: void Remove(Object\* dataSource);

[VB] Protected Friend Dim Sub Remove(ByVal dataSource As Object)

[JScript] package function Remove(dataSource : Object);

## Description

Deletes the **System.Windows.Forms.BindingManagerBase** associated with the specified data source. The data source associated with the **System.Windows.Forms.BindingManagerBase** to remove.

### s) *RemoveCore*

[C#]      protected      virtual      void      RemoveCore(object      dataSource);

[C++]      protected:      virtual      void      RemoveCore(Object\*      dataSource);

[VB]      Overridable      Protected      Sub      RemoveCore(ByVal      dataSource      As      Object)

[JScript]      protected      function      RemoveCore(dataSource      :      Object);

## Description

### t) *ICollection.CopyTo*

[C#]      void      ICollection.CopyTo(Array      ar,      int      index);

[C++]      void      ICollection::CopyTo(Array\*      ar,      int      index);

[VB]      Sub      CopyTo(ByVal      ar      As      Array, ByVal      index      As      Integer) Implements  
ICollection.CopyTo

[JScript]      function      ICollection.CopyTo(ar : Array, index : int);

### u) *IEnumerable.GetEnumerator*

[C#]           IEnumerator           IEnumerable.GetEnumerator();

[C++]           IEnumerator\*           IEnumerable::GetEnumerator();

[VB]      Function      GetEnumerator()      As      IEnumerator      Implements

1 IEnumerable.GetEnumerator

2 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;

3 BindingManagerBase class (System.Windows.Forms)

4 a) ToString

7 *Description*

8 Manages all **System.Windows.Forms.Binding** objects that are bound to the  
9 same data source and data member. This class is abstract.

10 The **System.Windows.Forms.BindingManagerBase** enables the  
11 synchronization of data-bound controls on a Windows form that are bound to the  
12 same data source. (To simple-bind a control to a data source, add a  
13 **System.Windows.Forms.Binding** object to the control's

14 **System.Windows.Forms.ControlBindingsCollection** , which is accessed  
15 through the **System.Windows.Forms.Control.DataBindings** property). For  
16 example, suppose that a form contains two **System.Windows.Forms.TextBox**  
17 controls that are bound to the same data source but to different columns (The  
18 data source might be a **System.Data.DataTable** that contains customer  
19 names, while the columns might contain the first name and last names). The two  
20 controls must be synchronized in order to display the correct first and last names  
21 together for the same customer. The

22 **System.Windows.Forms.CurrencyManager** , which inherits from the  
23 **System.Windows.Forms.BindingManagerBase** class, accomplishes this  
24 synchronization by maintaining a pointer to the current item for the list. The  
25 **System.Windows.Forms.TextBox** controls are bound to the current item so  
they display the information for the same row. When the current item changes,  
the **System.Windows.Forms.CurrencyManager** notifies all the bound  
controls so that they can refresh their data. Furthermore, you can set the  
**System.Windows.Forms.BindingManagerBase.Position** property to specify  
the row in the **System.Data.DataTable** that the controls are point to. To  
determine how many rows exist in the list, use the  
**System.Windows.Forms.BindingManagerBase.Count** property.

23 b) ToString

25 [C#] protected EventHandler onCurrentChangedHandler;

```

1  [C++]      protected:      EventHandler*      onCurrentChangedHandler;
2  [VB]      Protected      onCurrentChangedHandler      As      EventHandler
3  [JScript]  protected      var      onCurrentChangedHandler      :      EventHandler;
4

```

#### *Description*

Specifies the event handler for the **System.Windows.Forms.BindingManagerBase.CurrentChanged** event.

#### *c) ToString*

```

9  [C#]      protected      EventHandler      onPositionChangedHandler;
10 [C++]      protected:      EventHandler*      onPositionChangedHandler;
11 [VB]      Protected      onPositionChangedHandler      As      EventHandler
12 [JScript]  protected      var      onPositionChangedHandler      :      EventHandler;
13

```

#### *Description*

Specifies the event handler for the **System.Windows.Forms.BindingManagerBase.PositionChanged** event.

#### *d) BindingManagerBase*

*Example Syntax:*

#### *e) ToString*

```

21 [C#]      public      BindingManagerBase();
22 [C++]      public:      BindingManagerBase();
23 [VB]      Public      Sub      New()
24 [JScript]  public      function      BindingManagerBase();
25

```

## Description

Initializes a new instance of the **System.Windows.Forms.BindingManagerBase** class.

*f) Bindings*

*g) ToString*

```
[C#]      public      BindingsCollection      Bindings      {get;}
```

```
[C++]     public:     __property      BindingsCollection*      get_Bindings();
```

```
[VB]     Public  ReadOnly  Property  Bindings  As  BindingsCollection
```

```
[JScript] public  function  get  Bindings()  :  BindingsCollection;
```

## Description

Gets the collection of bindings being managed.

*h) Count*

*i) ToString*

```
[C#]      public      abstract      int      Count      {get;}
```

```
[C++]     public:     __property      virtual      int      get_Count()      =      0;
```

```
[VB]     MustOverride  Public  ReadOnly  Property  Count  As  Integer
```

```
[JScript] public  abstract  function  get  Count()  :  int;
```

## Description

When overridden in a derived class, gets the number of rows managed by the **System.Windows.Forms.BindingManagerBase**.

Use the **System.Windows.Forms.BindingManagerBase.Count** property to determine the last item in the list of rows maintained by the **System.Windows.Forms.BindingManagerBase**. To go to the last item, set the **System.Windows.Forms.BindingManagerBase.Position** property to the **System.Windows.Forms.BindingManagerBase.Count** property value minus one.

j) *Current*

k) *ToString*

```
[C#]      public      abstract      object      Current      {get;}
```

```
[C++]    public:  __property  virtual  Object*  get_Current()  =  0;
```

```
[VB]     MustOverride  Public  ReadOnly  Property  Current  As  Object
```

```
[JScript] public  abstract  function  get  Current()  :  Object;
```

### Description

When overridden in a derived class, gets the current object.

The **System.Windows.Forms.BindingManagerBase.Current** object contains the value of the current item in the data source. To use the value of the current item, you must cast the item to the **System.Type** of the object contained by the **System.Windows.Forms.Binding.DataSource**. For example, a **System.Data.DataTable** contains **System.Data.DataRowView** objects. To determine the type of the current object, use the **System.Object.GetType** and **System.Type.ToString** methods.

l) *Position*

m) *ToString*

```
[C#]      public      abstract      int      Position      {get;      set;}
```

```
[C++]    public:  __property  virtual  int  get_Position() = 0; public:  __property  virtual
```

```
void      set_Position(int)      =      0;
```

```
[VB]     MustOverride  Public  Property  Position  As  Integer
```



[JScript] public abstract function get Position() : int; public abstract function set Position(int);

#### *Description*

When overridden in a derived class, gets or sets the position in the underlying list that controls bound to this data source point to.

Use the **System.Windows.Forms.BindingManagerBase.Position** to iterate through the underlying list maintained by the **System.Windows.Forms.BindingManagerBase**. To go to the first item, set the **System.Windows.Forms.BindingManagerBase.Position** to zero. To go to the end of the list, set the **System.Windows.Forms.BindingManagerBase.Position** to the value of the **System.Windows.Forms.BindingManagerBase.Count** property minus one.

#### *n) ToString*

[C#]	public	event	EventHandler	CurrentChanged;
[C++]	public:	__event	EventHandler*	CurrentChanged;
[VB]	Public	Event	CurrentChanged	As EventHandler

#### *Description*

Occurs when the bound value changes.

For more information about handling events, see .

#### *o) ToString*

[C#]	public	event	EventHandler	PositionChanged;
[C++]	public:	__event	EventHandler*	PositionChanged;
[VB]	Public	Event	PositionChanged	As EventHandler

## Description

Occurs when the **System.Windows.Forms.BindingManagerBase.Position** has changed.

For more information about handling events, see .

### p) *AddNew*

[C#]	public	abstract	void	AddNew();
[C++]	public:	virtual	void	AddNew() = 0;
[VB]	MustOverride	Public	Sub	AddNew()
[JScript]	public	abstract	function	AddNew();

## Description

When overridden in a derived class, adds a new item to the underlying list.

### q) *CancelCurrentEdit*

[C#]	public	abstract	void	CancelCurrentEdit();
[C++]	public:	virtual	void	CancelCurrentEdit() = 0;
[VB]	MustOverride	Public	Sub	CancelCurrentEdit()
[JScript]	public	abstract	function	CancelCurrentEdit();

## Description

When overridden in a derived class, cancels the current edit.

This method is supported only if the data source implements the **System.ComponentModel.IEditableObject** interface. If the object does not implement the **System.ComponentModel.IEditableObject** interface, changes made to the data will not be discarded.

r) *EndCurrentEdit*

```
[C#]      public      abstract      void      EndCurrentEdit();
[C++]     public:     virtual     void      EndCurrentEdit()      =      0;
[VB]      MustOverride      Public      Sub      EndCurrentEdit()
[JScript]      public      abstract      function      EndCurrentEdit();
```

*Description*

When overridden in a derived class, ends the current edit.

This method is supported only if the data source implements the **System.ComponentModel.IEditableObject** interface. If the object does not implement the **System.ComponentModel.IEditableObject** interface, changes made to the data will not be discarded.

s) *GetItemProperties*

```
[C#]      public      abstract      PropertyDescriptorCollection      GetItemProperties();
[C++]     public:     virtual      PropertyDescriptorCollection*      GetItemProperties()      =      0;
[VB]      MustOverride      Public      Function      GetItemProperties()      As
PropertyDescriptorCollection
[JScript]      public      abstract      function      GetItemProperties()      :
PropertyDescriptorCollection; Get or sets the collection of property descriptors for
the binding.
```

*Description*

When overridden in a derived class, gets the collection of property descriptors for the binding.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** that represents the property descriptors for the binding.

t) *GetItemProperties*

```
[C#]      protected      internal      virtual      PropertyDescriptorCollection
GetItemProperties(ArrayList      dataSources,      ArrayList      listAccessors);

[C++]      protected      public:      virtual      PropertyDescriptorCollection*
GetItemProperties(ArrayList*      dataSources,      ArrayList*      listAccessors);

[VB] Overridable Protected Friend Dim Function GetItemProperties(ByVal
dataSources As ArrayList, ByVal listAccessors As ArrayList) As
PropertyDescriptorCollection

[JScript] package function GetItemProperties(dataSources : ArrayList,
listAccessors : ArrayList) : PropertyDescriptorCollection;
```

*Description*

Gets the collection of property descriptors for the binding using the specified **System.Collections.ArrayList**.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** that represents the property descriptors for the binding.

This method is used by developers creating data-bound controls. An **System.Collections.ArrayList** containing the data sources. An **System.Collections.ArrayList** containing the table's bound properties.

u) *GetItemProperties*

```
[C#] protected virtual PropertyDescriptorCollection GetItemProperties(Type
listType, int offset, ArrayList dataSources, ArrayList listAccessors);

[C++] protected: virtual PropertyDescriptorCollection* GetItemProperties(Type*
listType, int offset, ArrayList* dataSources, ArrayList* listAccessors);

[VB] Overridable Protected Function GetItemProperties(ByVal listType As Type,
```

```

1 ByVal offset As Integer, ByVal dataSources As ArrayList, ByVal listAccessors
2 As ArrayList) As PropertyDescriptorCollection
3 [JScript] protected function GetItemProperties(listType : Type, offset : int,
4 dataSources : ArrayList, listAccessors : ArrayList) : PropertyDescriptorCollection;
5

```

### *Description*

Gets the collection of property descriptors for the items managed by this **System.Windows.Forms.BindingManagerBase**.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** that represents the property descriptors for the binding.

This overload is used by developers creating data-bound controls. The **System.Type** of the bound list. A counter used to recursively call the method. An **System.Collections.ArrayList** containing the data sources. An **System.Collections.ArrayList** containing the table's bound properties.

### *v) GetListName*

```

14 [C#] protected internal abstract string GetListName(ArrayList listAccessors);
15 [C++] protected public: virtual String* GetListName(ArrayList* listAccessors) =
16 0;
17 [VB] MustOverride Protected Friend Dim Function GetListName(ByVal
18 listAccessors As ArrayList) As String
19 [JScript] package abstract function GetListName(listAccessors : ArrayList) :
20 String;
21

```

### *Description*

When overridden in a derived class, gets the name of the list supplying the data for the binding.

*Return Value:* The name of the list supplying the data for the binding. An **System.Collections.ArrayList** containing the table's bound properties.

## w) *OnCurrentChanged*

```
[C#] protected internal abstract void OnCurrentChanged(EventArgs e);
[C++] protected public: virtual void OnCurrentChanged(EventArgs* e) = 0;
[VB] MustOverride Protected Friend Dim Sub OnCurrentChanged(ByVal e As
EventArgs)
[JScript] package abstract function OnCurrentChanged(e : EventArgs);
```

### *Description*

When overridden in a derived class, raises the **System.Windows.Forms.BindingManagerBase.CurrentChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

## x) *PullData*

[C#]	protected	void	PullData();
[C++]	protected:	void	PullData();
[VB]	Protected	Sub	PullData()
[JScript]	protected	function	PullData();

### *Description*

Pulls data from the data-bound control into the data source.

## y) *PushData*

[C#]	protected	void	PushData();
[C++]	protected:	void	PushData();

[VB]	Protected	Sub	PushData()
[JScript]	protected	function	PushData();

#### Description

Pushes data from the data source into the data-bound control.

#### z) RemoveAt

[C#]	public	abstract	void	RemoveAt(int index);
[C++]	public:	virtual	void	RemoveAt(int index) = 0;
[VB]	MustOverride	Public	Sub	RemoveAt(ByVal index As Integer)
[JScript]	public	abstract	function	RemoveAt(index : int);

#### Description

When overridden in a derived class, deletes the row at the specified index from the underlying list.

The

**System.Windows.Forms.BindingManagerBase.RemoveAt(System.Int32)** method relies on the underlying data source to determine how the method behaves. (See the **System.Windows.Forms.Binding** class for a list of supported data sources). For classes that implement **System.Collections.IList**, **System.ComponentModel.IBindingList**, **System.ComponentModel.ITypedList**, as well as strongly typed classes that implement **System.Collections.IList**, the **System.Windows.Forms.BindingManagerBase.RemoveAt(System.Int32)** method actually deletes the row in the underlying list instead of removing it. The index of the row to delete.

#### aa) ResumeBinding

[C#]	public	abstract	void	ResumeBinding();
[C++]	public:	virtual	void	ResumeBinding() = 0;

[VB]	MustOverride	Public	Sub	ResumeBinding()
[JScript]	public	abstract	function	ResumeBinding();

#### Description

When overridden in a derived class, resumes data binding.

**System.Windows.Forms.BindingManagerBase.SuspendBinding** and **System.Windows.Forms.BindingManagerBase.ResumeBinding** are two methods that allow the temporary suspension and resumption of data binding. You would typically suspend data binding if the user must be allowed to make several edits to data fields before validation occurs. For example, if one field must be changed in accordance with a second, but where validating the first field would cause the second field to be in error.

#### bb) SuspendBinding

[C#]	public	abstract	void	SuspendBinding();
[C++]	public:	virtual	void	SuspendBinding() = 0;
[VB]	MustOverride	Public	Sub	SuspendBinding()
[JScript]	public	abstract	function	SuspendBinding();

#### Description

When overridden in a derived class, suspends data binding.

**System.Windows.Forms.BindingManagerBase.SuspendBinding** and **System.Windows.Forms.BindingManagerBase.ResumeBinding** are two methods that allow the temporary suspension and resumption of data binding. You would typically suspend data binding if the user must be allowed to make several edits to data fields before validation occurs. For example, if one field must be changed in accordance with a second, but where validating the first field would cause the second field to be in error.



### cc) *UpdateIsBinding*

[C#]	protected	abstract	void	UpdateIsBinding();
[C++]	protected:	virtual	void	UpdateIsBinding() = 0;
[VB]	MustOverride	Protected	Sub	UpdateIsBinding()
[JScript]	protected	abstract	function	UpdateIsBinding();

#### *Description*

When overridden in a derived class, updates the binding.

BindingMemberInfo structure (System.Windows.Forms)

### a) *UpdateIsBinding*

#### *Description*

Contains information that enables a **System.Windows.Forms.Binding** to resolve a data binding to either the property of an object or the property of the current object in a list of objects.

The **System.Windows.Forms.BindingMemberInfo** is returned by the **System.Windows.Forms.Binding.BindingMemberInfo** property of the **System.Windows.Forms.Binding** class.

### b) *BindingMemberInfo*

*Example Syntax:*

### c) *UpdateIsBinding*

[C#]	public	BindingMemberInfo(string	dataMember);
[C++]	public:	BindingMemberInfo(String*	dataMember);
[VB]	Public	Sub	New(ByVal dataMember As String)

[JScript] public function BindingMemberInfo(dataMember : String);

### Description

Initializes a new instance of the **System.Windows.Forms.BindingMemberInfo** class.

A **System.Windows.Forms.BindingMemberInfo** object is created automatically when you call the **System.Windows.Forms.Binding.#ctor** constructor with a control-property name, data source, and navigation path. The *dataMember* parameter contains the **System.Windows.Forms.BindingMemberInfo.BindingMember** string. A navigation path that resolves to either the property of an object or the property of the current object in a list of objects.

d) **BindingField**

e) **UpdateIsBinding**

[C#] public string BindingField {get;}

[C++] public: \_\_property String\* get\_BindingField();

[VB] Public ReadOnly Property BindingField As String

[JScript] public function get BindingField() : String;

### Description

Gets the data-bound object's property name.

The **System.Windows.Forms.BindingMemberInfo.BindingField** is the last item found in the navigation path returned by the **System.Windows.Forms.BindingMemberInfo.BindingMember** property. For example, if the navigation path is "Customers.custToOrders.OrderDate", the **System.Windows.Forms.BindingMemberInfo.BindingField** returns "OrderDate" which names the data-bound property of the data source.

1        *f)      BindingMember*

2        *g)      UpdateIsBinding*

3  
4    [C#]            public            string            BindingMember            {get;}  
5    [C++]          public:          \_\_property          String\*          get\_BindingMember();  
6    [VB]    Public    ReadOnly    Property    BindingMember    As    String  
7    [JScript]    public    function    get    BindingMember()    :    String;

8  
9    *Description*

10 Gets the information that is used to specify the data-bound object's property name.

11 A **System.Windows.Forms.BindingMemberInfo** object is created  
12 automatically when you call the **System.Windows.Forms.Binding.#ctor**  
13 constructor with a control-property name, data source, and navigation path. The  
14 *dataMember* parameter contains the  
15 **System.Windows.Forms.BindingMemberInfo.BindingMember** string.

14        *h)      BindingPath*

15        *i)      UpdateIsBinding*

16  
17  
18    [C#]            public            string            BindingPath            {get;}  
19    [C++]          public:          \_\_property          String\*          get\_BindingPath();  
20    [VB]    Public    ReadOnly    Property    BindingPath    As    String  
21    [JScript]    public    function    get    BindingPath()    :    String;

22  
23    *Description*

24 Gets the property name, or the period-delimited hierarchy of property names,  
25 that precedes the data-bound object's property name.

j) *Equals*

[C#] public override bool Equals(object otherObject);  
[C++] public: bool Equals(Object\* otherObject);  
[VB] Overrides Public Function Equals(ByVal otherObject As Object) As Boolean  
[JScript] public override function Equals(otherObject : Object) : Boolean;

*Description*

k) *GetHashCode*

[C#] public override int GetHashCode();  
[C++] public: int GetHashCode();  
[VB] Overrides Public Function GetHashCode() As Integer  
[JScript] public override function GetHashCode() : int;

*Description*

BindingsCollection class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a collection of **System.Windows.Forms.Binding** objects for a control.

Simple data binding is accomplished by adding **System.Windows.Forms.Binding** objects to a **System.Windows.Forms.BindingsCollection**. Any object that inherits from the **System.Windows.Forms.Control** class can access the **System.Windows.Forms.BindingsCollection** through the **System.Windows.Forms.Control.DataBindings** property. For a list of Windows controls that support data binding, see the **System.Windows.Forms.Binding** class.

*b) Count*

*c) ToString*

[C#]	public	override	int	Count	{get;}
[C++]	public:	__property	virtual	int	get_Count();
[VB]	Overrides	Public	ReadOnly	Property	Count As Integer
[JScript]	public	function	get	Count()	: int;

#### *Description*

Gets the total number of bindings in the collection.

*d) IsReadOnly*

*e) IsSynchronized*

*f) Item*

*g) ToString*

#### *Description*

Gets the **System.Windows.Forms.Binding** at the specified index. The index of the **System.Windows.Forms.Binding** to find.

h) *List*

i) *ToString*

[C#] protected override ArrayList List {get;}

[C++] protected: \_\_property virtual ArrayList\* get\_List();

[VB] Overrides Protected ReadOnly Property List As ArrayList

[JScript] protected function get List() : ArrayList;

#### *Description*

Gets the bindings in the collection as an object.

j) *SyncRoot*

k) *ToString*

#### *Description*

Occurs when the collection has changed.

For more information about handling events, see .

l) *Add*

[C#] protected internal void Add(Binding binding);

[C++] protected public: void Add(Binding\* binding);

[VB] Protected Friend Dim Sub Add(ByVal binding As Binding)

[JScript] package function Add(binding : Binding);

#### *Description*

Adds the specified binding to the collection. The **System.Windows.Forms.Binding** to add to the collection.

*m) AddCore*

[C#]      protected      virtual      void      AddCore(Binding      dataBinding);  
 [C++]      protected:      virtual      void      AddCore(Binding\*      dataBinding);  
 [VB]      Overridable      Protected      Sub      AddCore(ByVal      dataBinding      As      Binding)  
 [JScript]      protected      function      AddCore(dataBinding      :      Binding);

*Description*

Adds a **System.Windows.Forms.Binding** to the collection.

*n) Clear*

[C#]              protected              internal              void              Clear();  
 [C++]              protected              public:              void              Clear();  
 [VB]              Protected              Friend              Dim              Sub              Clear()  
 [JScript]              package              function              Clear();

*Description*

*o) ClearCore*

[C#]              protected              virtual              void              ClearCore();  
 [C++]              protected:              virtual              void              ClearCore();  
 [VB]              Overridable              Protected              Sub              ClearCore()

[JScript]                      protected                      function                      ClearCore();

*Description*

Clears the collection of any members.

*p)      OnCollectionChanged*

[C#] protected virtual void OnCollectionChanged(CollectionChangeEventArgs  
ccevent);

[C++] protected: virtual void OnCollectionChanged(CollectionChangeEventArgs\*  
ccevent);

[VB] Overridable Protected Sub OnCollectionChanged(ByVal ccevent As  
CollectionChangeEventArgs)

[JScript]              protected              function              OnCollectionChanged(ccevent              :  
CollectionChangeEventArgs);

*Description*

Raises the  
**System.Windows.Forms.BindingsCollection.CollectionChanged** event.

Raising an event invokes the event handler through a delegate. For more  
information, see . A **System.ComponentModel.CollectionChangeEventArgs**  
that contains the event data.

*q)      Remove*

[C#]              protected              internal              void              Remove(Binding              binding);

[C++]              protected              public:              void              Remove(Binding\*              binding);

[VB] Protected Friend Dim Sub Remove(ByVal binding As Binding)



1 [JScript] package function Remove(binding : Binding);

3 *Description*

4 Deletes the specified binding from the collection. The Binding to remove from the collection.

5  
6 *r) RemoveAt*

7 [C#] protected internal void RemoveAt(int index);

8 [C++] protected public: void RemoveAt(int index);

9 [VB] Protected Friend Dim Sub RemoveAt(ByVal index As Integer)

10 [JScript] package function RemoveAt(index : int);

12  
13 *Description*

14 Deletes the binding from the collection at the specified index. The index of the **System.Windows.Forms.Binding** to remove.

15  
16 *s) RemoveCore*

17 [C#] protected virtual void RemoveCore(Binding dataBinding);

18 [C++] protected: virtual void RemoveCore(Binding\* dataBinding);

19 [VB] Overridable Protected Sub RemoveCore(ByVal dataBinding As Binding)

20 [JScript] protected function RemoveCore(dataBinding : Binding);

22  
23 *Description*

24 Removes the specified **System.Windows.Forms.Binding** from the collection. The **System.Windows.Forms.Binding** object to remove.

*t) ShouldSerializeMyAll*

```
[C#]      protected      internal      bool      ShouldSerializeMyAll();
[C++]     protected      public:       bool      ShouldSerializeMyAll();
[VB] Protected Friend Dim Function ShouldSerializeMyAll() As Boolean
[JScript] package      function      ShouldSerializeMyAll()      :      Boolean;
```

*Description*

Gets a value that indicates whether the collection should be serialized.

*Return Value:* **true** if the collection count is greater than zero; otherwise, **false** .

BootMode enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the mode the computer was started in.

This enumeration is used by the **System.Windows.Forms.SystemInformation.BootMode** property in the **System.Windows.Forms.SystemInformation** class.

*b) ToString*

```
[C#]      public      const      BootMode      FailSafe;
[C++]     public:     const      BootMode      FailSafe;
[VB] Public Const FailSafe As BootMode
[JScript] public      var      FailSafe      :      BootMode;
```

*Description*

The computer was started by using only the basic files and drivers.

*c) ToString*

[C#]	public	const	BootMode	FailSafeWithNetwork;
[C++]	public:	const	BootMode	FailSafeWithNetwork;
[VB]	Public	Const	FailSafeWithNetwork	As BootMode
[JScript]	public	var	FailSafeWithNetwork	: BootMode;

*Description*

The computer was started by using the basic files, drivers, and services necessary to start networking.

*d) ToString*

[C#]	public	const	BootMode	Normal;
[C++]	public:	const	BootMode	Normal;
[VB]	Public	Const	Normal	As BootMode
[JScript]	public	var	Normal	: BootMode;

*Description*

The computer was started in standard mode.

## Border3DSide enumeration (System.Windows.Forms)

### a) *ToString*

#### *Description*

Specifies the sides of a rectangle to apply a three-dimensional border to.

Use the members of this enumeration with the **System.Windows.Forms.ControlPaint.DrawBorder3D(System.Drawing.Graphics, System.Drawing.Rectangle)** method of the **System.Windows.Forms.ControlPaint** class.

### b) *ToString*

[C#]	public	const	Border3DSide	All;
[C++]	public:	const	Border3DSide	All;
[VB]	Public	Const	All	As Border3DSide
[JScript]	public	var	All	: Border3DSide;

#### *Description*

A three-dimensional border on all four edges and fill the middle of the rectangle with the color defined for three-dimensional controls.

### c) *ToString*

[C#]	public	const	Border3DSide	Bottom;
[C++]	public:	const	Border3DSide	Bottom;
[VB]	Public	Const	Bottom	As Border3DSide
[JScript]	public	var	Bottom	: Border3DSide;

*Description*

A three-dimensional border on the bottom side of the rectangle.

*d) ToString*

[C#]	public	const	Border3DSide	Left;
[C++]	public:	const	Border3DSide	Left;
[VB]	Public	Const	Left	As Border3DSide
[JScript]	public	var	Left	: Border3DSide;

*Description*

A three-dimensional border on the left edge of the control.

*e) ToString*

[C#]	public	const	Border3DSide	Middle;
[C++]	public:	const	Border3DSide	Middle;
[VB]	Public	Const	Middle	As Border3DSide
[JScript]	public	var	Middle	: Border3DSide;

*Description*

The interior of the rectangle is filled with the color defined for three-dimensional controls instead of the background color for the form.

*f) ToString*

[C#]	public	const	Border3DSide	Right;
------	--------	-------	--------------	--------

[C++]	public:	const	Border3DSide	Right;
[VB]	Public	Const	Right	As Border3DSide
[JScript]	public	var	Right	: Border3DSide;

#### Description

A three-dimensional border on the right side of the rectangle.

#### g) ToString

[C#]	public	const	Border3DSide	Top;
[C++]	public:	const	Border3DSide	Top;
[VB]	Public	Const	Top	As Border3DSide
[JScript]	public	var	Top	: Border3DSide;

#### Description

A three-dimensional border on the top edge of the rectangle.

Border3DStyle enumeration (System.Windows.Forms)

#### a) ToString

#### Description

Specifies the style of a three-dimensional border.

Use the members of this enumeration when calling the **System.Windows.Forms.ControlPaint.DrawBorder3D(System.Drawing.Graphics,System.Drawing.Rectangle)** method of the **System.Windows.Forms.ControlPaint** class.

**b) ToString**

[C#]	public	const	Border3DStyle	Adjust;
[C++]	public:	const	Border3DStyle	Adjust;
[VB]	Public	Const	Adjust As	Border3DStyle
[JScript]	public	var	Adjust :	Border3DStyle;

*Description*

The border is drawn outside the specified rectangle, preserving the dimensions of the rectangle for drawing.

**c) ToString**

[C#]	public	const	Border3DStyle	Bump;
[C++]	public:	const	Border3DStyle	Bump;
[VB]	Public	Const	Bump As	Border3DStyle
[JScript]	public	var	Bump :	Border3DStyle;

*Description*

The inner and outer edges of the border have a raised appearance.

**d) ToString**

[C#]	public	const	Border3DStyle	Etched;
[C++]	public:	const	Border3DStyle	Etched;
[VB]	Public	Const	Etched As	Border3DStyle
[JScript]	public	var	Etched :	Border3DStyle;

*Description*

The inner and outer edges of the border have an etched appearance.

*e) ToString*

[C#]	public	const	Border3DStyle	Flat;
[C++]	public:	const	Border3DStyle	Flat;
[VB]	Public	Const	Flat	As Border3DStyle
[JScript]	public	var	Flat	: Border3DStyle;

*Description*

The border has no three-dimensional effects.

*f) ToString*

[C#]	public	const	Border3DStyle	Raised;
[C++]	public:	const	Border3DStyle	Raised;
[VB]	Public	Const	Raised	As Border3DStyle
[JScript]	public	var	Raised	: Border3DStyle;

*Description*

The border has raised inner and outer edges.

*g) ToString*

[C#]	public	const	Border3DStyle	RaisedInner;
[C++]	public:	const	Border3DStyle	RaisedInner;



[VB]	Public	Const	RaisedInner	As	Border3DStyle
[JScript]	public	var	RaisedInner	:	Border3DStyle;

*Description*

The border has a raised inner edge and no outer edge.

***h) ToString***

[C#]	public	const	Border3DStyle		RaisedOuter;
[C++]	public:	const	Border3DStyle		RaisedOuter;
[VB]	Public	Const	RaisedOuter	As	Border3DStyle
[JScript]	public	var	RaisedOuter	:	Border3DStyle;

*Description*

The border has a raised outer edge and no inner edge.

***i) ToString***

[C#]	public	const	Border3DStyle		Sunken;
[C++]	public:	const	Border3DStyle		Sunken;
[VB]	Public	Const	Sunken	As	Border3DStyle
[JScript]	public	var	Sunken	:	Border3DStyle;

*Description*

The border has sunken inner and outer edges.

**j) ToString**

[C#]	public	const	Border3DStyle	SunkenInner;
[C++]	public:	const	Border3DStyle	SunkenInner;
[VB]	Public	Const	SunkenInner	As Border3DStyle
[JScript]	public	var	SunkenInner	: Border3DStyle;

*Description*

The border has a sunken inner edge and no outer edge.

**k) ToString**

[C#]	public	const	Border3DStyle	SunkenOuter;
[C++]	public:	const	Border3DStyle	SunkenOuter;
[VB]	Public	Const	SunkenOuter	As Border3DStyle
[JScript]	public	var	SunkenOuter	: Border3DStyle;

*Description*

The border has a sunken outer edge and no inner edge.

BorderStyle enumeration (System.Windows.Forms)

**a) ToString**

*Description*

Specifies the border style for a control.

Use the members of this enumeration to set the border style for controls that have a changeable border.

**b) ToString**

[C#]	public	const	BorderStyle	Fixed3D;
[C++]	public:	const	BorderStyle	Fixed3D;
[VB]	Public	Const	Fixed3D	As BorderStyle
[JScript]	public	var	Fixed3D	: BorderStyle;

*Description*

A three-dimensional border.

**c) ToString**

[C#]	public	const	BorderStyle	FixedSingle;
[C++]	public:	const	BorderStyle	FixedSingle;
[VB]	Public	Const	FixedSingle	As BorderStyle
[JScript]	public	var	FixedSingle	: BorderStyle;

*Description*

A single-line border.

**d) ToString**

[C#]	public	const	BorderStyle	None;
[C++]	public:	const	BorderStyle	None;
[VB]	Public	Const	None	As BorderStyle
[JScript]	public	var	None	: BorderStyle;

*Description*

No border.

BoundsSpecified enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the bounds of the control to use when defining a control's size and position.

Use the members of this enumeration when calling the **System.Windows.Forms.Control.SetBoundsCore(System.Int32,System.Int32,System.Int32,System.Int32,System.Windows.Forms.BoundsSpecified)** and **System.Windows.Forms.Control.SetBounds(System.Int32,System.Int32,System.Int32,System.Int32)** methods of the **System.Windows.Forms.Control** class.

*b) ToString*

[C#]	public	const	BoundsSpecified	All;
[C++]	public:	const	BoundsSpecified	All;
[VB]	Public	Const	All	As BoundsSpecified
[JScript]	public	var	All	: BoundsSpecified;

*Description*

Both **System.Windows.Forms.Control.Location** and **System.Windows.Forms.Control.Size** property values are defined.

c) *ToString*

[C#]	public	const	BoundsSpecified	Height;
[C++]	public:	const	BoundsSpecified	Height;
[VB]	Public	Const	Height As	BoundsSpecified
[JScript]	public	var	Height :	BoundsSpecified;

*Description*

Specifies the height of the control is defined.

d) *ToString*

[C#]	public	const	BoundsSpecified	Location;
[C++]	public:	const	BoundsSpecified	Location;
[VB]	Public	Const	Location As	BoundsSpecified
[JScript]	public	var	Location :	BoundsSpecified;

*Description*

Both **X** and **Y** coordinates of the control are defined.

e) *ToString*

[C#]	public	const	BoundsSpecified	None;
[C++]	public:	const	BoundsSpecified	None;
[VB]	Public	Const	None As	BoundsSpecified
[JScript]	public	var	None :	BoundsSpecified;

*Description*

No bounds are specified.

*f) ToString*

[C#]	public	const	BoundsSpecified	Size;
------	--------	-------	-----------------	-------

[C++]	public:	const	BoundsSpecified	Size;
-------	---------	-------	-----------------	-------

[VB]	Public	Const	Size	As	BoundsSpecified
------	--------	-------	------	----	-----------------

[JScript]	public	var	Size	:	BoundsSpecified;
-----------	--------	-----	------	---	------------------

*Description*

Both **System.Windows.Forms.Control.Width** and **System.Windows.Forms.Control.Height** property values of the control are defined.

*g) ToString*

[C#]	public	const	BoundsSpecified	Width;
------	--------	-------	-----------------	--------

[C++]	public:	const	BoundsSpecified	Width;
-------	---------	-------	-----------------	--------

[VB]	Public	Const	Width	As	BoundsSpecified
------	--------	-------	-------	----	-----------------

[JScript]	public	var	Width	:	BoundsSpecified;
-----------	--------	-----	-------	---	------------------

*Description*

Specifies the width of the control is defined.

***h) ToString***

[C#]	public	const		BoundsSpecified	X;
[C++]	public:	const		BoundsSpecified	X;
[VB]	Public	Const	X	As	BoundsSpecified
[JScript]	public	var	X	:	BoundsSpecified;

***Description***

Specifies the left edge of the control is defined.

***i) ToString***

[C#]	public	const		BoundsSpecified	Y;
[C++]	public:	const		BoundsSpecified	Y;
[VB]	Public	Const	Y	As	BoundsSpecified
[JScript]	public	var	Y	:	BoundsSpecified;

***Description***

Specifies the top edge of the control is defined.

Button class (System.Windows.Forms)

***a) ToString***

***Description***

Represents a Windows button control.

A **System.Windows.Forms.Button** may be clicked by using the mouse or the ENTER key if the button has focus.

**b) Button**

*Example Syntax:*

**c) ToString**

[C#]	public	Button();
[C++]	public:	Button();
[VB]	Public	Sub New()
[JScript]	public	function Button();

*Description*

Initializes a new instance of the **System.Windows.Forms.Button** class.

By default the **System.Windows.Forms.Button** displays no caption. To specify the caption text, set the **System.Windows.Forms.Control.Text** property.



- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *Bottom*
- o) *Bounds*
- p) *CanFocus*
- q) *CanSelect*
- r) *Capture*
- s) *CausesValidation*
- t) *ClientRectangle*
- u) *ClientSize*
- v) *CompanyName*
- w) *Container*
- x) *ContainsFocus*
- y) *ContextMenu*
- z) *Controls*

- aa) *Created*
- bb) *CreateParams*
- cc) *ToString*

#### *Description*

This is called when creating a window. Inheriting classes can override this to add extra functionality, but should not forget to first call `base.CreateParams()` to make sure the control continues to work correctly.

- dd) *Cursor*
- ee) *DataBindings*
- ff) *DefaultImeMode*
- gg) *DefaultSize*
- hh) *DesignMode*
- ii) *DialogResult*
- jj) *ToString*

#### *Description*

Gets or sets a value that is returned to the parent form when the button is clicked.

If the value of this property is set to anything other than **None** , and if the parent form was displayed through the **System.Windows.Forms.Form.ShowDialog** method, clicking the button closes the parent form without your having to hook up any events. The form's **System.Windows.Forms.Form.DialogResult** property is then set to the **System.Windows.Forms.Button.DialogResult** of the button when the button is clicked.

1	<i>kk) DisplayRectangle</i>
2	<i>ll) Disposing</i>
3	<i>mm) Dock</i>
4	<i>nn) Enabled</i>
5	<i>oo) Events</i>
6	<i>pp) FlatStyle</i>
7	<i>qq) Focused</i>
8	<i>rr) Font</i>
9	<i>ss) FontHeight</i>
10	<i>tt) ForeColor</i>
11	<i>uu) Handle</i>
12	<i>vv) HasChildren</i>
13	<i>ww) Height</i>
14	<i>xx) Image</i>
15	<i>yy) ImageAlign</i>
16	<i>zz) ImageIndex</i>
17	<i>aaa) ImageList</i>
18	<i>bbb) ImeMode</i>
19	<i>ccc) InvokeRequired</i>
20	<i>ddd) IsAccessible</i>
21	<i>eee) IsDefault</i>
22	<i>fff) IsDisposed</i>
23	<i>ggg) IsHandleCreated</i>
24	
25	

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

- hhh) Left*
- iii) Location*
- jjj) Name*
- kkk) Parent*
- lll) ProductName*
- mmm) ProductVersion*
- nnn) RecreatingHandle*
- ooo) Region*
- ppp) RenderRightToLeft*
- qqq) ResizeRedraw*
- rrr) Right*
- sss) RightToLeft*
- ttt) ShowFocusCues*
- uuu) ShowKeyboardCues*
- vvv) Site*
- www) Size*
- xxx) TabIndex*
- yyy) TabStop*
- zzz) Tag*
- aaaa) Text*
- bbbb) TextAlign*
- cccc) Top*
- dddd) TopLevelControl*

*eeee) Visible*

*ffff) Width*

*gggg) WindowTarget*

*hhhh) ToString*

*iiii) NotifyDefault*

[C#]        public        virtual        void        NotifyDefault(bool        value);

[C++]        public:        virtual        void        NotifyDefault(bool        value);

[VB]    Overridable    Public    Sub    NotifyDefault(ByVal    value    As    Boolean)

[JScript]        public        function        NotifyDefault(value        :        Boolean);

### *Description*

Notifies the **System.Windows.Forms.Button** whether it is the default button so that it can adjust its appearance accordingly.

This method is called by the parent form to notify the **System.Windows.Forms.Button** that it should be set as the default button and to allow it to adjust its appearance accordingly. Typically, a button that is the default button for a form has a thicker border than other buttons on the form. **true** if the button is to have the appearance of the default button; otherwise, **false**.

*jjjj) OnClick*

[C#]        protected        override        void        OnClick(EventArgs        e);

[C++]        protected:        void        OnClick(EventArgs\*        e);

[VB]    Overrides    Protected    Sub    OnClick(ByVal    e    As    EventArgs)

[JScript]        protected        override        function        OnClick(e        :        EventArgs);

## Description

This method actually raises the Click event. Inheriting classes should override this if they wish to be notified of a Click event. (This is far preferable to actually adding an event handler.) They should not, however, forget to call `base.OnClick(e)`; before exiting, to ensure that other recipients do actually get the event. This event has no additional information besides the sender.

### kkkk) OnMouseUp

[C#]   protected   override   void   OnMouseUp(MouseEventArgs   mevent);

[C++]   protected:   void   OnMouseUp(MouseEventArgs\*   mevent);

[VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)

[JScript] protected override function OnMouseUp(mevent : MouseEventArgs);

## Description

Raises the **System.Windows.Forms.ButtonBase.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event. A **System.Windows.Forms.MouseEventArgs** that contains the event data.

### llll) PerformClick

[C#]                   public                   void                   PerformClick();

[C++]           public:           \_\_sealed           void           PerformClick();

[VB]           NotOverridable           Public           Sub           PerformClick()

[JScript]           public           function           PerformClick();

## Description

Generates a **System.Windows.Forms.Control.Click** event for a button.

This method can be called to raise the **System.Windows.Forms.Control.Click** event.

#### *mmm)ProcessMnemonic*

[C#]      protected      override      bool      ProcessMnemonic(char      charCode);  
[C++]      protected:      bool      ProcessMnemonic(\_\_wchar\_t      charCode);  
[VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)  
As      Boolean  
[JScript] protected override function ProcessMnemonic(charCode : Char) :  
Boolean;

#### *Description*

Lets a control process mnmemonic characters. Inheriting classes can override this to add extra functionality, but should not forget to call base.ProcessMnemonic(charCode); to ensure basic functionality remains unchanged.

*Return Value:* boolean a boolean indicating whether the mnmemonic was processsed [consumed] or not. The mnemonic character entered.

#### *nnnn) ToString*

[C#]      public      override      string      ToString();  
[C++]      public:      String\*      ToString();  
[VB]      Overrides      Public      Function      ToString()      As      String  
[JScript]      public      override      function      ToString()      :      String;

#### *Description*

Provides some interesting information for the Button control in String form.

*Return Value:* A String.

## 0000) *WndProc*

```
[C#]    protected    override    void    WndProc(ref    Message    m);
[C++]    protected:    void    WndProc(Message*    m);
[VB]    Overrides    Protected    Sub    WndProc(ByRef    m    As    Message)
[JScript]    protected    override    function    WndProc(m    :    Message);
```

### *Description*

The button's window procedure. Inheriting classes can override this to add extra functionality, but should not forget to call `base.wndProc(m)`; to ensure the button continues to function properly. A Windows Message Object.

ButtonBase class (System.Windows.Forms)

### *a) WndProc*

### *Description*

Implements the basic functionality common to button controls.

You do not typically inherit from **System.Windows.Forms.ButtonBase** . To create your own button class, inherit from the **System.Windows.Forms.Button** , **System.Windows.Forms.CheckBox** , or **System.Windows.Forms.RadioButton** class.

### *b) ButtonBase*

### *Example Syntax:*

### *c) WndProc*

```
[C#]                                protected                                ButtonBase();
[C++]                                protected:                                ButtonBase();
```



1	[VB]	Protected	Sub	New()
2	[JScript]	protected	function	ButtonBase();

3

4 *Description*

5 Initializes a new instance of the **System.Windows.Forms.ButtonBase** class.

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *Bottom*
- o) *Bounds*
- p) *CanFocus*
- q) *CanSelect*
- r) *Capture*
- s) *CausesValidation*
- t) *ClientRectangle*
- u) *ClientSize*
- v) *CompanyName*
- w) *Container*
- x) *ContainsFocus*
- y) *ContextMenu*
- z) *Controls*

- aa) *Created*
- bb) *CreateParams*
- cc) *WndProc*

#### *Description*

- dd) *Cursor*
- ee) *DataBindings*
- ff) *DefaultImeMode*
- gg) *WndProc*

#### *Description*

Gets the default Input Method Editor(IME) mode supported by this control.

As implemented in the **System.Windows.Forms.ButtonBase** class, this property always returns the **System.Windows.Forms.ImeMode.Disable** value.

- hh) *DefaultSize*
- ii) *WndProc*

[C#]	protected	override	Size	DefaultSize	{get;}
[C++]	protected:	__property	virtual	Size	get_DefaultSize();
[VB]	Overrides	Protected	ReadOnly	Property	DefaultSize As Size
[JScript]	protected	function	get	DefaultSize()	: Size;

## Description

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

*jj) DesignMode*

*kk) DisplayRectangle*

*ll) Disposing*

*mm) Dock*

*nn) Enabled*

*oo) Events*

*pp) FlatStyle*

*qq) WndProc*

## Description

Gets or sets the flat style appearance of the button control.

When the **System.Windows.Forms.ButtonBase.FlatStyle** property of the **System.Windows.Forms.RadioButton** and **System.Windows.Forms.CheckBox** classes is set to **System.Windows.Forms.FlatStyle.System**, the control is drawn by the user's operating system and the check alignment is based upon the **CheckAlign** and **System.Windows.Forms.ButtonBase.TextAlign** property values. The **CheckAlign** property value is not changed, but the appearance of the control can be affected. The check box is horizontally aligned with either the left or right edge of the control (a left or center alignment appears left aligned, right remains unchanged), and vertically aligned the same as the descriptive text. For example, if you have a **System.Windows.Forms.CheckBox** control with a **CheckAlign** property value of **System.Drawing.ContentAlignment.MiddleCenter** and a **System.Windows.Forms.ButtonBase.TextAlign** property value of **System.Drawing.ContentAlignment.TopRight**, and the **System.Windows.Forms.ButtonBase.FlatStyle** property value is set to

**System.Windows.Forms.FlatStyle.System** , the check box alignment will appear to be **System.Drawing.ContentAlignment.TopLeft** while the text alignment remains unchanged.

*rr) Focused*

*ss) Font*

*tt) FontHeight*

*uu) ForeColor*

*vv) Handle*

*ww) HasChildren*

*xx) Height*

*yy) Image*

*zz) WndProc*

### *Description*

Gets or sets the image that is displayed on a button control.

When the **System.Windows.Forms.ButtonBase.Image** property is set, the **System.Windows.Forms.ButtonBase.ImageList** property will be set to **null** , and the **System.Windows.Forms.ButtonBase.ImageIndex** property will be set to its default, -1.

*aaa) ImageAlign*

*bbb) WndProc*

```
[C#]      public      ContentAlignment      ImageAlign      {get;      set;}
```

```
[C++] public: __property ContentAlignment get_ImageAlign();public: __property
```

```
void      set_ImageAlign(ContentAlignment);
```

```

1  [VB]      Public      Property      ImageAlign      As      ContentAlignment
2  [JScript] public function get ImageAlign() : ContentAlignment;public function set
3  ImageAlign(ContentAlignment);

```

#### *Description*

Gets or sets the alignment of the image on the button control.

*ccc) ImageIndex*

*ddd) WndProc*

```

10 [C#]      public      int      ImageIndex      {get;      set;}
11 [C++] public: __property int get_ImageIndex();public: __property void
12 set_ImageIndex(int);

```

```

13 [VB]      Public      Property      ImageIndex      As      Integer
14 [JScript] public function get ImageIndex() : int;public function set
15 ImageIndex(int);

```

#### *Description*

Gets or sets the image list index value of the image displayed on the button control.

When the **System.Windows.Forms.ButtonBase.ImageIndex** or **System.Windows.Forms.ButtonBase.ImageList** properties are set, the **System.Windows.Forms.ButtonBase.Image** property is set to its default value, **null** .

eee) *ImageList*

fff) *WndProc*

```
[C#]      public      ImageList      ImageList      {get;      set;}
[C++] public: __property ImageList* get_ImageList();public: __property void
set_ImageList(ImageList*);
[VB]      Public      Property      ImageList      As      ImageList
[JScript] public function get ImageList() : ImageList;public function set
ImageList(ImageList);
```

#### *Description*

Gets or sets the **System.Windows.Forms.ImageList** that contains the **System.Drawing.Image** displayed on a button control.

When the **System.Windows.Forms.ButtonBase.ImageList** or **System.Windows.Forms.ButtonBase.ImageIndex** property is set the **System.Windows.Forms.ButtonBase.Image** property will be set to its default value, **null**.

ggg) *ImeMode*

hhh) *WndProc*

```
[C#]      public      new      ImeMode      ImeMode      {get;      set;}
[C++] public: __property ImeMode get_ImeMode();public: __property void
set_ImeMode(ImeMode);
[VB]      Public      Property      ImeMode      As      ImeMode
[JScript] public function get ImeMode() : ImeMode;public function set
ImeMode(ImeMode);
```

*Description*

Gets or sets the Input Method Editor (IME) mode supported by this control.

*iii) InvokeRequired*

*jjj) IsAccessible*

*kkk) IsDefault*

*lll) WndProc*

*Description*

Gets or sets a value indicating whether the button control is the default button.



mmm) *IsDisposed*

nnn) *IsHandleCreated*

ooo) *Left*

ppp) *Location*

qqq) *Name*

rrr) *Parent*

sss) *ProductName*

ttt) *ProductVersion*

uuu) *RecreatingHandle*

vvv) *Region*

www) *RenderRightToLeft*

xxx) *ResizeRedraw*

yyy) *Right*

zzz) *RightToLeft*

aaaa) *ShowFocusCues*

bbbb) *ShowKeyboardCues*

cccc) *Site*

dddd) *Size*

eeee) *TabIndex*

ffff) *TabStop*

gggg) *Tag*

hhhh) *Text*

iiii) *TextAlign*

*jjjj) WndProc*

*Description*

Gets or sets the alignment of the text on the button control.

*kkkk) Top*

*llll) TopLevelControl*

*mmmm)Visible*

*nnnn) Width*

*oooo) WindowTarget*

*pppp) CreateAccessibilityInstance*

[C#] protected override AccessibleObject CreateAccessibilityInstance();

[C++] protected: AccessibleObject\* CreateAccessibilityInstance();

[VB] Overrides Protected Function CreateAccessibilityInstance() As

AccessibleObject

[JScript] protected override function CreateAccessibilityInstance() :

AccessibleObject;

*Description*

*qqqq) Dispose*

[C#] protected override void Dispose(bool disposing);

```

1 [C++]      protected:      void      Dispose(bool      disposing);
2 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)
3 [JScript] protected override function Dispose(disposing : Boolean);

```

#### *Description*

Removes hooks from the parent, and then calls the base class to perform all other cleanup

#### *rrrr) OnEnabledChanged*

```

9 [C#]      protected      override      void      OnEnabledChanged(EventArgs      e);
10 [C++]      protected:      void      OnEnabledChanged(EventArgs*      e);
11 [VB] Overrides Protected Sub OnEnabledChanged(ByVal e As EventArgs)
12 [JScript] protected override function OnEnabledChanged(e : EventArgs);

```

#### *Description*

#### *ssss) OnGotFocus*

```

18 [C#]      protected      override      void      OnGotFocus(EventArgs      e);
19 [C++]      protected:      void      OnGotFocus(EventArgs*      e);
20 [VB] Overrides Protected Sub OnGotFocus(ByVal e As EventArgs)
21 [JScript] protected override function OnGotFocus(e : EventArgs);

```

#### *Description*

Raises the

**System.Windows.Forms.ButtonBase.OnGotFocus(System.EventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains the event data.

*tttt) OnKeyDown*

[C#] protected override void OnKeyDown(KeyEventArgs kevent);

[C++] protected: void OnKeyDown(KeyEventArgs\* kevent);

[VB] Overrides Protected Sub OnKeyDown(ByVal kevent As KeyEventArgs)

[JScript] protected override function OnKeyDown(kevent : KeyEventArgs);

#### *Description*

Raises the

**System.Windows.Forms.ButtonBase.OnKeyUp(System.Windows.Forms.KeyEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see XXX. A **System.Windows.Forms.KeyEventArgs** that contains the event data.

*uuuu) OnKeyUp*

[C#] protected override void OnKeyUp(KeyEventArgs kevent);

[C++] protected: void OnKeyUp(KeyEventArgs\* kevent);

[VB] Overrides Protected Sub OnKeyUp(ByVal kevent As KeyEventArgs)

[JScript] protected override function OnKeyUp(kevent : KeyEventArgs);

#### *Description*

Raises the  
**System.Windows.Forms.ButtonBase.OnKeyUp(System.Windows.Forms.KeyEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see XXX. A **System.Windows.Forms.KeyEventArgs** that contains the event data.

*vvvv) OnLostFocus*

[C#]      protected      override      void      OnLostFocus(EventArgs      e);

[C++]      protected:      void      OnLostFocus(EventArgs\*      e);

[VB]      Overrides      Protected      Sub      OnLostFocus(ByVal      e      As      EventArgs)

[JScript]      protected      override      function      OnLostFocus(e      :      EventArgs);

#### *Description*

Raises the  
**System.Windows.Forms.ButtonBase.OnLostFocus(System.EventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see XXX. An **System.EventArgs** that contains the event data.

*www)OnMouseDown*

[C#]      protected      override      void      OnMouseDown(MouseEventArgs      mevent);

[C++]      protected:      void      OnMouseDown(MouseEventArgs\*      mevent);

[VB]      Overrides      Protected      Sub      OnMouseDown(ByVal      mevent      As      MouseEventArgs)

[JScript]      protected      override      function      OnMouseDown(mevent : MouseEventArgs);

#### *Description*

Raises the

**System.Windows.Forms.Control.OnMouseDown(System.Windows.Forms.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

*xxxx) OnMouseEnter*

[C#] protected override void OnMouseEnter(EventArgs eventargs);

[C++] protected: void OnMouseEnter(EventArgs\* eventargs);

[VB] Overrides Protected Sub OnMouseEnter(ByVal eventargs As EventArgs)

[JScript] protected override function OnMouseEnter(eventargs : EventArgs);

#### *Description*

Raises the

**System.Windows.Forms.Control.OnMouseEnter(System.EventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains the event data.

*yyyy) OnMouseLeave*

[C#] protected override void OnMouseLeave(EventArgs eventargs);

[C++] protected: void OnMouseLeave(EventArgs\* eventargs);

[VB] Overrides Protected Sub OnMouseLeave(ByVal eventargs As EventArgs)

[JScript] protected override function OnMouseLeave(eventargs : EventArgs);

#### *Description*

Raises the

**System.Windows.Forms.Control.OnMouseLeave(System.EventArgs)**  
event.

Raising an event invokes the event handler through a delegate. For an overview,  
see . An **System.EventArgs** that contains the event data.

*zzzz) OnMouseMove*

[C#] protected override void OnMouseMove(MouseEventArgs mevent);

[C++] protected: void OnMouseMove(MouseEventArgs\* mevent);

[VB] Overrides Protected Sub OnMouseMove(ByVal mevent As  
MouseEventArgs)

[JScript] protected override function OnMouseMove(mevent : MouseEventArgs);

#### *Description*

Raises the

**System.Windows.Forms.Control.OnMouseMove(System.Windows.Forms.  
s.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview,  
see . A **System.Windows.Forms.MouseEventArgs** that contains the event  
data.

*aaaaa) OnMouseUp*

[C#] protected override void OnMouseUp(MouseEventArgs mevent);

[C++] protected: void OnMouseUp(MouseEventArgs\* mevent);

[VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)

[JScript] protected override function OnMouseUp(mevent : MouseEventArgs);

#### *Description*

Raises the

**System.Windows.Forms.ButtonBase.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see XXX. A **System.Windows.Forms.MouseEventArgs** that contains the event data.

#### *bbbbbb) OnPaint*

[C#]      protected      override      void      OnPaint(PaintEventArgs      pevent);

[C++]      protected:      void      OnPaint(PaintEventArgs\*      pevent);

[VB] Overrides Protected Sub OnPaint(ByVal pevent As PaintEventArgs)

[JScript] protected override function OnPaint(pevent : PaintEventArgs);

#### *Description*

Raises the

**System.Windows.Forms.ButtonBase.OnPaint(System.Windows.Forms.PaintEventArgs)** event.

Raising an event invokes the event handler through a delegate. For an overview, see XXX. A **System.Windows.Forms.PaintEventArgs** that contains the event data.

#### *cccccc) OnParentChanged*

[C#]      protected      override      void      OnParentChanged(EventArgs      e);

[C++]      protected:      void      OnParentChanged(EventArgs\*      e);

[VB] Overrides Protected Sub OnParentChanged(ByVal e As EventArgs)

[JScript] protected override function OnParentChanged(e : EventArgs);

#### *Description*



### ***dddd) OnTextChanged***

```
[C#]    protected    override    void    OnTextChanged(EventArgs    e);
[C++]    protected:    void    OnTextChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnTextChanged(ByVal e As EventArgs)
[JScript] protected override function OnTextChanged(e : EventArgs);
```

#### ***Description***

### ***eeee) OnVisibleChanged***

```
[C#]    protected    override    void    OnVisibleChanged(EventArgs    e);
[C++]    protected:    void    OnVisibleChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)
[JScript] protected override function OnVisibleChanged(e : EventArgs);
```

#### ***Description***

### ***ffff) ResetFlagsandPaint***

```
[C#]                protected                void                ResetFlagsandPaint();
[C++]                protected:                void                ResetFlagsandPaint();
[VB]                Protected                Sub                ResetFlagsandPaint()
[JScript]            protected            function            ResetFlagsandPaint();
```

## Description

Used for quick re-painting of the button after the pressed state.

### *ggggg) WndProc*

[C#]      protected      override      void      WndProc(ref      Message      m);

[C++]      protected:      void      WndProc(Message\*      m);

[VB]      Overrides      Protected      Sub      WndProc(ByRef      m      As      Message)

[JScript] protected override function WndProc(m : Message);

ButtonBase.ButtonBaseAccessibleObject class (System.Windows.Forms)

### *a)      WndProc*

## Description

### *b)      ButtonBase.ButtonBaseAccessibleObject*

*Example Syntax:*

### *c)      WndProc*

[C#]      public      ButtonBase.ButtonBaseAccessibleObject(Control      owner);

[C++]      public:      ButtonBaseAccessibleObject(Control\*      owner);

[VB]      Public      Sub      New(ByVal      owner      As      Control)

[JScript] public function ButtonBase.ButtonBaseAccessibleObject(owner :

Control);

*Description*

- d) *Bounds*
- e) *DefaultAction*
- f) *Description*
- g) *Handle*
- h) *Help*
- i) *KeyboardShortcut*
- j) *Name*
- k) *Owner*
- l) *Parent*
- m) *Role*
- n) *State*
- o) *Value*
- p) *DoDefaultAction*

[C#]	public	override	void	DoDefaultAction();
[C++]	public:		void	DoDefaultAction();
[VB]	Overrides	Public	Sub	DoDefaultAction()
[JScript]	public	override	function	DoDefaultAction();

*Description*

1        ButtonBorderStyle enumeration (System.Windows.Forms)

2        *a)        UseStdAccessibleObjects*

5        *Description*

6        Specifies the border style for a button control.

7        This enumeration is used by members such as  
8        **System.Windows.Forms.ControlPaint.DrawBorder(System.Drawing.Gra**  
9        **phics,System.Drawing.Rectangle,System.Drawing.Color,System.Windo**  
10       **ws.Forms.ButtonBorderStyle) .**

11       *b)        UseStdAccessibleObjects*

12       [C#]        public        const        ButtonBorderStyle       Dashed;  
13       [C++]       public:       const       ButtonBorderStyle       Dashed;  
14       [VB]       Public       Const       Dashed       As       ButtonBorderStyle  
15       [JScript]   public       var       Dashed       :       ButtonBorderStyle;

17       *Description*

18       A dashed border.

19       *c)        UseStdAccessibleObjects*

20  
21       [C#]        public        const        ButtonBorderStyle       Dotted;  
22       [C++]       public:       const       ButtonBorderStyle       Dotted;  
23       [VB]       Public       Const       Dotted       As       ButtonBorderStyle  
24       [JScript]   public       var       Dotted       :       ButtonBorderStyle;

*Description*

A dotted-line border.

*d) UseStdAccessibleObjects*

[C#]	public	const	ButtonBorderStyle	Inset;
[C++]	public:	const	ButtonBorderStyle	Inset;
[VB]	Public	Const	Inset As	ButtonBorderStyle
[JScript]	public	var	Inset :	ButtonBorderStyle;

*Description*

A sunken border.

*e) UseStdAccessibleObjects*

[C#]	public	const	ButtonBorderStyle	None;
[C++]	public:	const	ButtonBorderStyle	None;
[VB]	Public	Const	None As	ButtonBorderStyle
[JScript]	public	var	None :	ButtonBorderStyle;

*Description*

No border.

*f) UseStdAccessibleObjects*

[C#]	public	const	ButtonBorderStyle	Outset;
[C++]	public:	const	ButtonBorderStyle	Outset;

1 [VB] Public Const Outset As ButtonBorderStyle

2 [JScript] public var Outset : ButtonBorderStyle;

3  
4 *Description*

5 A raised border.

6 **g) UseStdAccessibleObjects**

7  
8 [C#] public const ButtonBorderStyle Solid;

9 [C++] public: const ButtonBorderStyle Solid;

10 [VB] Public Const Solid As ButtonBorderStyle

11 [JScript] public var Solid : ButtonBorderStyle;

12  
13 *Description*

14 A solid border.

15 ButtonState enumeration (System.Windows.Forms)

16 **a) ToString**

17  
18  
19 *Description*

20 Specifies the appearance of a button.

21 **b) ToString**

22  
23 [C#] public const ButtonState All;

24 [C++] public: const ButtonState All;

25 [VB] Public Const All As ButtonState

[JScript]	public	var	All	:	ButtonState;
-----------	--------	-----	-----	---	--------------

### Description

All flags except **Normal** are set.

#### c) ToString

[C#]	public	const	ButtonState	Checked;
------	--------	-------	-------------	----------

[C++]	public:	const	ButtonState	Checked;
-------	---------	-------	-------------	----------

[VB]	Public	Const	Checked	As	ButtonState
------	--------	-------	---------	----	-------------

[JScript]	public	var	Checked	:	ButtonState;
-----------	--------	-----	---------	---	--------------

### Description

The button has a checked or latched appearance. Use this appearance to show that a toggle button has been pressed.

#### d) ToString

[C#]	public	const	ButtonState	Flat;
------	--------	-------	-------------	-------

[C++]	public:	const	ButtonState	Flat;
-------	---------	-------	-------------	-------

[VB]	Public	Const	Flat	As	ButtonState
------	--------	-------	------	----	-------------

[JScript]	public	var	Flat	:	ButtonState;
-----------	--------	-----	------	---	--------------

### Description

The button has a flat, two-dimensional appearance.

*e) ToString*

[C#]	public	const	ButtonState	Inactive;
[C++]	public:	const	ButtonState	Inactive;
[VB]	Public	Const	Inactive	As ButtonState
[JScript]	public	var	Inactive	: ButtonState;

*Description*

The button is inactive (grayed).

*f) ToString*

[C#]	public	const	ButtonState	Normal;
[C++]	public:	const	ButtonState	Normal;
[VB]	Public	Const	Normal	As ButtonState
[JScript]	public	var	Normal	: ButtonState;

*Description*

The button has its normal appearance (three-dimensional).

*g) ToString*

[C#]	public	const	ButtonState	Pushed;
[C++]	public:	const	ButtonState	Pushed;
[VB]	Public	Const	Pushed	As ButtonState
[JScript]	public	var	Pushed	: ButtonState;



*Description*

The button appears pressed.

CaptionButton enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the type of caption button to display.

This enumeration is used by members such as

**System.Windows.Forms.ControlPaint.DrawCaptionButton(System.Drawing.Graphics, System.Drawing.Rectangle, System.Windows.Forms.CaptionButton, System.Windows.Forms.ButtonState)** .

*b) ToString*

[C#]	public	const	CaptionButton	Close;
[C++]	public:	const	CaptionButton	Close;
[VB]	Public	Const	Close As	CaptionButton
[JScript]	public	var	Close :	CaptionButton;

*Description*

A Close button.

*c) ToString*

[C#]	public	const	CaptionButton	Help;
[C++]	public:	const	CaptionButton	Help;

[VB]	Public	Const	Help	As	CaptionButton
[JScript]	public	var	Help	:	CaptionButton;

*Description*

A Help button.

*d) ToString*

[C#]	public	const	CaptionButton	Maximize;	
[C++]	public:	const	CaptionButton	Maximize;	
[VB]	Public	Const	Maximize	As	CaptionButton
[JScript]	public	var	Maximize	:	CaptionButton;

*Description*

A Maximize button.

*e) ToString*

[C#]	public	const	CaptionButton	Minimize;	
[C++]	public:	const	CaptionButton	Minimize;	
[VB]	Public	Const	Minimize	As	CaptionButton
[JScript]	public	var	Minimize	:	CaptionButton;

*Description*

A Minimize button.

*f) ToString*

[C#]	public	const	CaptionButton	Restore;
[C++]	public:	const	CaptionButton	Restore;
[VB]	Public	Const	Restore	As CaptionButton
[JScript]	public	var	Restore	: CaptionButton;

*Description*

A Restore button.

CharacterCasing enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the case of characters in a **System.Windows.Forms.TextBox** control.

Use the members of this enumeration to set the value of the **System.Windows.Forms.TextBox.CharacterCasing** property of the **System.Windows.Forms.TextBox** control.

*b) ToString*

[C#]	public	const	CharacterCasing	Lower;
[C++]	public:	const	CharacterCasing	Lower;
[VB]	Public	Const	Lower	As CharacterCasing
[JScript]	public	var	Lower	: CharacterCasing;

*Description*

Converts all characters to lowercase.

*c) ToString*

[C#]	public	const	CharacterCasing	Normal;
[C++]	public:	const	CharacterCasing	Normal;
[VB]	Public	Const	Normal	As CharacterCasing
[JScript]	public	var	Normal	: CharacterCasing;

*Description*

The case of characters is left unchanged.

*d) ToString*

[C#]	public	const	CharacterCasing	Upper;
[C++]	public:	const	CharacterCasing	Upper;
[VB]	Public	Const	Upper	As CharacterCasing
[JScript]	public	var	Upper	: CharacterCasing;

*Description*

Converts all characters to 'uppercase.

CheckBox class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a Windows check box.

Use a **System.Windows.Forms.CheckBox** to give the user an option, such as true/false or yes/no. The check box control can display an image or text or both.

b) *CheckBox*

*Example Syntax:*

c) *ToString*

[C#]	public	CheckBox();
[C++]	public:	CheckBox();
[VB]	Public	Sub New()
[JScript]	public	function CheckBox();

*Description*

Initializes a new instance of the **System.Windows.Forms.CheckBox** class.

By default, when a new **System.Windows.Forms.CheckBox** is instantiated, **System.Windows.Forms.CheckBox.AutoCheck** is set to **true** , **System.Windows.Forms.CheckBox.Checked** is set to **false** , and **System.Windows.Forms.CheckBox.Appearance** is set to **Normal** .

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *Appearance*
- l) *ToString*

#### *Description*

Gets or sets the value that determines the appearance of a check box control.

If **System.Windows.Forms.CheckBox.Appearance** value is set to **Appearance.Normal** , the check box has a typical appearance. If the value is set to **Button** , the check box appears like a toggle button, which may be toggled to an up or down state.

- m) *AutoCheck*
- n) *ToString*

```
[C#]          public          bool          AutoCheck          {get;          set;}

[C++] public: __property bool get_AutoCheck();public: __property void
set_AutoCheck(bool);

[VB]          Public          Property          AutoCheck          As          Boolean

[JScript] public function get AutoCheck() : Boolean;public function set
```

AutoCheck(Boolean);

*Description*

Gets or set a value indicating whether the **System.Windows.Forms.CheckBox.Checked** or **System.Windows.Forms.CheckBox.CheckState** values and the check box's appearance are automatically changed when the check box is clicked.

If **System.Windows.Forms.CheckBox.AutoCheck** is set to false, you will need to add code to update the **System.Windows.Forms.CheckBox.Checked** or **System.Windows.Forms.CheckBox.CheckState** values in the **System.Windows.Forms.Control.Click** event handler.

*o) BackColor*

*p) BackgroundImage*

*q) BindingContext*

*r) Bottom*

*s) Bounds*

*t) CanFocus*

*u) CanSelect*

*v) Capture*

*w) CausesValidation*

*x) CheckAlign*

*y) ToString*

*Description*

Gets or sets the horizontal and vertical alignment of a check box on a check box control.

z) *Checked*

aa) *ToString*

```
[C#]      public      bool      Checked      {get;      set;}
```

```
[C++] public: __property bool get_Checked();public: __property void  
set_Checked(bool);
```

```
[VB]      Public      Property      Checked      As      Boolean
```

```
[JScript] public function get Checked() : Boolean;public function set  
Checked(Boolean);
```

### *Description*

Gets or set a value indicating whether the check box is in the checked state.

When the value is **true** , the check box portion of the control displays a check mark. If the **System.Windows.Forms.CheckBox.Appearance** property is set to **Button** , the control will appear sunken when

**System.Windows.Forms.CheckBox.Checked** is **true** and raised like a standard button when **false** .

bb) *CheckState*

cc) *ToString*

```
[C#]      public      CheckState      CheckState      {get;      set;}
```

```
[C++] public: __property CheckState get_CheckState();public: __property void  
set_CheckState(CheckState);
```

```
[VB]      Public      Property      CheckState      As      CheckState
```

```
[JScript] public function get CheckState() : CheckState;public function set
```



1 CheckState(CheckState);

3 *Description*

4 Gets or sets the state of the check box.

5 If the **System.Windows.Forms.CheckBox.ThreeState** property is set to  
6 **false** , the **System.Windows.Forms.CheckBox.CheckState** property value  
7 may only be set to **CheckState.Indeterminate** in code and not by user  
8 interaction.

9 *dd) ClientRectangle*

10 *ee) ClientSize*

11 *ff) CompanyName*

12 *gg) Container*

13 *hh) ContainsFocus*

14 *ii) ContextMenu*

15 *jj) Controls*

16 *kk) Created*

17 *ll) CreateParams*

18 *mm) ToString*

20 *Description*

21 Gets the information used to create the handle for the  
22 **System.Windows.Forms.CheckBox** control.

- nn) Cursor*
- oo) DataBindings*
- pp) DefaultImeMode*
- qq) DefaultSize*
- rr) ToString*

### *Description*

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

- ss) *DesignMode*
- tt) *DisplayRectangle*
- uu) *Disposing*
- vv) *Dock*
- ww) *Enabled*
- xx) *Events*
- yy) *FlatStyle*
- zz) *Focused*
- aaa) *Font*
- bbb) *FontHeight*
- ccc) *ForeColor*
- ddd) *Handle*
- eee) *HasChildren*
- fff) *Height*
- ggg) *Image*
- hhh) *ImageAlign*
- iii) *ImageIndex*
- jjj) *ImageList*
- kkk) *ImeMode*
- lll) *InvokeRequired*
- mmm) *IsAccessible*
- nnn) *IsDefault*
- ooo) *IsDisposed*

MSI-861US.APP  
509-324-9256  
lee@hayes.plc

1	<i>ppp) IsHandleCreated</i>
2	<i>qqq) Left</i>
3	<i>rrr) Location</i>
4	<i>sss) Name</i>
5	<i>ttt) Parent</i>
6	<i>uuu) ProductName</i>
7	<i>vvv) ProductVersion</i>
8	<i>www) RecreatingHandle</i>
9	<i>xxx) Region</i>
10	<i>yyy) RenderRightToLeft</i>
11	<i>zzz) ResizeRedraw</i>
12	<i>aaaa) Right</i>
13	<i>bbbb) RightToLeft</i>
14	<i>cccc) ShowFocusCues</i>
15	<i>dddd) ShowKeyboardCues</i>
16	<i>eeee) Site</i>
17	<i>ffff) Size</i>
18	<i>gggg) TabIndex</i>
19	<i>hhhh) TabStop</i>
20	<i>iiii) Tag</i>
21	<i>jjjj) Text</i>
22	<i>kkkk) TextAlign</i>
23	<i>llll) ToString</i>
24	
25	

### Description

Gets or sets a value indicating the alignment of the text on the checkbox control.

*mmmm)ThreeState*

*nnnn) ToString*

[C#]            public            bool            ThreeState            {get;            set;}

[C++]   public:   \_\_property   bool   get\_ThreeState();public:   \_\_property   void  
set\_ThreeState(bool);

[VB]            Public            Property            ThreeState            As            Boolean

[JScript]   public   function   get   ThreeState()   :   Boolean;public   function   set  
ThreeState(Boolean);

### Description

Gets or sets a value indicating whether the check box will allow three check states rather than two.

If the **System.Windows.Forms.CheckBox.ThreeState** property is set to **false** , the **System.Windows.Forms.CheckBox.CheckState** property value may only be set to **CheckState.Indeterminate** in code and not by user interaction.

oooo) *Top*

pppp) *TopLevelControl*

qqqq) *Visible*

rrrr) *Width*

ssss) *WindowTarget*

tttt) *ToString*

*Description*

Occurs when the value of the **System.Windows.Forms.CheckBox.Appearance** property changes.

For more information about handling events, see .

uuuu) *ToString*

*Description*

Occurs when the value of the **System.Windows.Forms.CheckBox.Checked** property changes.

For more information about handling events, see .

vvvv) *ToString*

[C#]	public	event	EventHandler	CheckStateChanged;
[C++]	public:	__event	EventHandler*	CheckStateChanged;
[VB]	Public	Event	CheckStateChanged	As EventHandler

## Description

Occurs when the value of the **System.Windows.Forms.CheckBox.CheckState** property changes.

For more information about handling events, see .

### *www)CreateAccessibilityInstance*

[C#] protected override AccessibleObject CreateAccessibilityInstance();

[C++] protected: AccessibleObject\* CreateAccessibilityInstance();

[VB] Overrides Protected Function CreateAccessibilityInstance() As AccessibleObject

[JScript] protected override function CreateAccessibilityInstance() : AccessibleObject;

## Description

Constructs the new instance of the accessibility object for this control. Subclasses should not call base.CreateAccessibilityObject.

### *xxxx) OnAppearanceChanged*

[C#] protected virtual void OnAppearanceChanged(EventArgs e);

[C++] protected: virtual void OnAppearanceChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnAppearanceChanged(ByVal e As EventArgs)

[JScript] protected function OnAppearanceChanged(e : EventArgs);

## Description

Raises the **System.Windows.Forms.CheckBox.AppearanceChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *yyyy) OnCheckedChanged*

[C#] protected virtual void OnCheckedChanged(EventArgs e);

[C++] protected: virtual void OnCheckedChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnCheckedChanged(ByVal e As EventArgs)

[JScript] protected function OnCheckedChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.CheckBox.CheckedChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *zzzz) OnCheckStateChanged*

[C#] protected virtual void OnCheckStateChanged(EventArgs e);

[C++] protected: virtual void OnCheckStateChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnCheckStateChanged(ByVal e As EventArgs)

[JScript] protected function OnCheckStateChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.CheckBox.CheckStateChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.



### *aaaaa) OnClick*

```
[C#]      protected      override      void      OnClick(EventArgs      e);
[C++]      protected:      void      OnClick(EventArgs*      e);
[VB] Overrides Protected Sub OnClick(ByVal e As EventArgs)
[JScript] protected override function OnClick(e : EventArgs);
```

#### *Description*

Fires the event indicating that the control has been clicked. Inheriting controls should use this in favour of actually listening to the event, but should not forget to call `base.OnClicked()` to ensure that the event is still fired for external listeners.

### *bbbbbb) OnHandleCreated*

```
[C#]      protected      override      void      OnHandleCreated(EventArgs      e);
[C++]      protected:      void      OnHandleCreated(EventArgs*      e);
[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

#### *Description*

We override this to ensure that the control's click values are set up correctly.

### *ccccc) OnMouseUp*

```
[C#]      protected      override      void      OnMouseUp(MouseEventArgs      mevent);
[C++]      protected:      void      OnMouseUp(MouseEventArgs*      mevent);
[VB] Overrides Protected Sub OnMouseUp(ByVal mevent As MouseEventArgs)
```

[JScript] protected override function OnMouseUp(mevent : MouseEventArgs);

### *Description*

Raises the **System.Windows.Forms.ButtonBase.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event.

### *dddd) ProcessMnemonic*

[C#] protected override bool ProcessMnemonic(char charCode);

[C++] protected: bool ProcessMnemonic(\_\_wchar\_t charCode);

[VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)

As Boolean

[JScript] protected override function ProcessMnemonic(charCode : Char) :

Boolean; Overridden to handle mnemonics properly.

### *eeee) ToString*

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

### *Description*

Provides some interesting information for the CheckBox control in String form.

CheckBox.CheckBoxAccessibleObject class (System.Windows.Forms)

a) *WndProc*

*Description*

b) *CheckBox.CheckBoxAccessibleObject*

*Example Syntax:*

c) *WndProc*

[C#] public CheckBox.CheckBoxAccessibleObject(Control owner);

[C++] public: CheckBoxAccessibleObject(Control\* owner);

[VB] Public Sub New(ByVal owner As Control)

[JScript] public function CheckBox.CheckBoxAccessibleObject(owner : Control);

*Description*

d) *Bounds*

e) *DefaultAction*

f) *WndProc*

*Description*

g) *Description*

h) *Handle*

i) *Help*

j) *KeyboardShortcut*

k) *Name*

l) *Owner*

m) *Parent*

n) *Role*

o) *WndProc*

*Description*

p) *State*

q) *WndProc*

[C#]      public      override      AccessibleStates      State      {get;}

[C++]    public:    \_\_property    virtual    AccessibleStates    get\_State();

[VB]    Overrides    Public    ReadOnly    Property    State    As    AccessibleStates

[JScript]    public    function    get    State()    :    AccessibleStates;

*Description*

*r) Value*

CheckedListBox.CheckedIndexCollection class (System.Windows.Forms)

*a) UseStdAccessibleObjects*

### *Description*

Encapsulates the collection of indexes of checked items (including items in an indeterminate state) in a **System.Windows.Forms.CheckedListBox**.

The checked indexes collection is a subset of the indexes into the collection of all items in the **System.Windows.Forms.CheckedListBox** control. These indexes specify items in a checked or indeterminate state.

*b) Count*

*c) UseStdAccessibleObjects*

[C#]	public	int	Count	{get;}
[C++]	public:	__property	int	get_Count();
[VB]	Public	ReadOnly	Property	Count As Integer
[JScript]	public	function	get	Count() : int;

### *Description*

Gets the number of checked items.

*d) IsReadOnly*

*e) UseStdAccessibleObjects*

[C#]	public	bool	IsReadOnly	{get;}
[C++]	public:	__property	bool	get_IsReadOnly();

```

1  [VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
2  [JScript]  public      function      get      IsReadOnly()      :      Boolean;

```

#### *Description*

Gets a value indicating whether the collection is read-only.

#### *f) Item*

#### *g) UseStdAccessibleObjects*

```

9  [C#]      public      int      this[int      index]      {get;}
10 [C++]      public:      __property      int      get_Item(int      index);
11 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Integer
12 [JScript]  returnValue      =      CheckedIndexCollectionObject.Item(index);

```

#### *Description*

Gets the index of a checked item in the **System.Windows.Forms.CheckedListBox** control.

The *index* parameter is an index into the checked indexes collection. The return value is the corresponding indexed value, which is the index of the checked item in the items collection. An index into the checked indexes collection. This index specifies the index of the checked item you want to retrieve.

#### *h) Contains*

```

21 [C#]      public      bool      Contains(int      index);
22 [C++]      public:      bool      Contains(int      index);
23 [VB] Public Function Contains(ByVal index As Integer) As Boolean
24 [JScript]  public      function      Contains(index      :      int)      :      Boolean;

```

## *Description*

Determines whether the specified index is located in the collection.

*Return Value:* **true** if the specified index from the **System.Windows.Forms.CheckedListBox.ObjectCollection** is an item in this collection; otherwise, **false** .

You can use this method to determine whether an index from the **System.Windows.Forms.CheckedListBox.Items** collection is in the **System.Windows.Forms.CheckedListBox.CheckedIndices** collection. The index to locate in the collection.

### *i) CopyTo*

[C#]        public        void        CopyTo(Array        dest,        int        index);

[C++]        public:        \_\_sealed        void        CopyTo(Array\*        dest,        int        index);

[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)

[JScript]        public        function        CopyTo(dest        :        Array,        index        :        int);

## *Description*

Copies the entire collection into an existing array at a specified location within the array.

You can use this method to combine the selected indexes from multiple collections into a single array. The destination array. The zero-based relative index in *dest* at which copying begins.

### *j) GetEnumerator*

[C#]        public        IEnumerator        GetEnumerator();

[C++]        public:        \_\_sealed        IEnumerator\*        GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript]      public      function      GetEnumerator()      :      IEnumerator;

### *Description*

Returns an enumerator that can be used to iterate through the

**System.Windows.Forms.CheckedListBox.CheckedIndices** collection.

*Return Value:* An **System.Collections.IEnumerator** for navigating through the list.

### *k)      IndexOf*

[C#]              public              int              IndexOf(int              index);

[C++]              public:              int              IndexOf(int              index);

[VB]      Public      Function      IndexOf(ByVal      index      As      Integer)      As      Integer

[JScript]      public      function      IndexOf(index      :      int)      :      int;

### *Description*

Returns an index into the collection of checked indexes.

*Return Value:* The index that specifies the index of the checked item or -1 if the *index* parameter is not in the checked indexes collection. For more information, see the examples in the

**System.Windows.Forms.CheckedListBox.CheckedIndexCollection** class overview.

The *index* parameter is the index of a checked item in the items collection. The return value is the corresponding index into the checked indexes collection. The index of the checked item.

### *l)      IList.Add*

[C#]              int              IList.Add(object              value);

[C++]              int              IList::Add(Object\*              value);



1 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

2 [JScript] function IList.Add(value : Object) : int;

3 **m) *IList.Clear***

4  
5 [C#] void IList.Clear();

6 [C++] void IList::Clear();

7 [VB] Sub Clear() Implements IList.Clear

8 [JScript] function IList.Clear();

9 **n) *IList.Contains***

10  
11 [C#] bool IList.Contains(object index);

12 [C++] bool IList::Contains(Object\* index);

13 [VB] Function Contains(ByVal index As Object) As Boolean Implements

14 IList.Contains

15 [JScript] function IList.Contains(index : Object) : Boolean;

16 **o) *IList.IndexOf***

17  
18 [C#] int IList.IndexOf(object index);

19 [C++] int IList::IndexOf(Object\* index);

20 [VB] Function IndexOf(ByVal index As Object) As Integer Implements

21 IList.IndexOf

22 [JScript] function IList.IndexOf(index : Object) : int;

**p) *IList.Insert***

[C#] void IList.Insert(int index, object value);

[C++] void IList::Insert(int index, Object\* value);

[VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements IList.Insert

[JScript] function IList.Insert(index : int, value : Object);

**q) *IList.Remove***

[C#] void IList.Remove(object value);

[C++] void IList::Remove(Object\* value);

[VB] Sub Remove(ByVal value As Object) Implements IList.Remove

[JScript] function IList.Remove(value : Object);

**r) *IList.RemoveAt***

[C#] void IList.RemoveAt(int index);

[C++] void IList::RemoveAt(int index);

[VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

[JScript] function IList.RemoveAt(index : int);

ListView.CheckedIndexCollection class (System.Windows.Forms)

**a) *ToString***

*Description*

**b) *ListView.CheckedIndexCollection***

*Example Syntax:*

**c) *ToString***

```
[C#]      public      ListView.CheckedIndexCollection(ListView      owner);
[C++]      public:      CheckedIndexCollection(ListView*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      ListView)
[JScript] public function ListView.CheckedIndexCollection(owner : ListView);
```

*Description*

**d) *Count***

**e) *ToString***

```
[C#]      public      int      Count      {get;}
[C++]      public:      __property      int      get_Count();
[VB]      Public      ReadOnly      Property      Count      As      Integer
[JScript] public      function      get      Count()      :      int;
```

*Description*

Gets the number of currently selected items.

**f) *IsReadOnly***

**g) *ToString***

```
[C#]      public      bool      IsReadOnly      {get;}
```

```

1  [C++]      public:      __property      bool      get_IsReadOnly();
2  [VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
3  [JScript]      public      function      get      IsReadOnly()      :      Boolean;

```

#### *Description*

Gets whether the collection is read-only.

**h) Item**

**i) ToString**

```

10 [C#]      public      int      this[int      index]      {get;}
11 [C++]      public:      __property      int      get_Item(int      index);
12 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Integer
13 [JScript]      returnValue      =      CheckedIndexCollectionObject.Item(index);

```

#### *Description*

Selected item in the list.

**j) Contains**

```

19 [C#]      public      bool      Contains(int      checkedIndex);
20 [C++]      public:      bool      Contains(int      checkedIndex);
21 [VB] Public Function Contains(ByVal checkedIndex As Integer) As Boolean
22 [JScript]      public      function      Contains(checkedIndex      :      int)      :      Boolean;

```

#### *Description*

**k) GetEnumerator**

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]        public:      __sealed      IEnumerator*      GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript]    public      function      GetEnumerator()      :      IEnumerator;
```

*Description*

**l) IndexOf**

```
[C#]          public          int          IndexOf(int          checkedIndex);
[C++]        public:          int          IndexOf(int          checkedIndex);
[VB] Public Function IndexOf(ByVal checkedIndex As Integer) As Integer
[JScript]    public      function      IndexOf(checkedIndex      :      int)      :      int;
```

*Description*

**m) ICollection.CopyTo**

```
[C#]          void          ICollection.CopyTo(Array          dest,          int          index);
[C++]        void          ICollection::CopyTo(Array*          dest,          int          index);
[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo
[JScript] function ICollection.CopyTo(dest : Array, index : int);
```

***n) IList.Add***

[C#] int IList.Add(object value);  
[C++] int IList::Add(Object\* value);  
[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add  
[JScript] function IList.Add(value : Object) : int;

***o) IList.Clear***

[C#] void IList.Clear();  
[C++] void IList::Clear();  
[VB] Sub Clear() Implements IList.Clear  
[JScript] function IList.Clear();

***p) IList.Contains***

[C#] bool IList.Contains(object checkedIndex);  
[C++] bool IList::Contains(Object\* checkedIndex);  
[VB] Function Contains(ByVal checkedIndex As Object) As Boolean Implements  
IList.Contains  
[JScript] function IList.Contains(checkedIndex : Object) : Boolean;

***q) IList.IndexOf***

[C#] int IList.IndexOf(object checkedIndex);  
[C++] int IList::IndexOf(Object\* checkedIndex);  
[VB] Function IndexOf(ByVal checkedIndex As Object) As Integer Implements

1 IList.IndexOf

2 [JScript] function IList.IndexOf(index : Object) : int;

3 **r) IList.Insert**

4  
5 [C#] void IList.Insert(int index, object value);

6 [C++] void IList::Insert(int index, Object\* value);

7 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

8 IList.Insert

9 [JScript] function IList.Insert(index : int, value : Object);

10 **s) IList.Remove**

11  
12 [C#] void IList.Remove(object value);

13 [C++] void IList::Remove(Object\* value);

14 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

15 [JScript] function IList.Remove(value : Object);

16 **t) IList.RemoveAt**

17  
18 [C#] void IList.RemoveAt(int index);

19 [C++] void IList::RemoveAt(int index);

20 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

21 [JScript] function IList.RemoveAt(index : int);

CheckedListBox.CheckedItemCollection class (System.Windows.Forms)

a) *ToString*

*Description*

Encapsulates the collection of checked items (including items in an indeterminate state) in a **System.Windows.Forms.CheckedListBox** control.

The checked items collection is a subset of all items in the **System.Windows.Forms.CheckedListBox** control; it contains only those items that are in a checked or indeterminate state.

b) *Count*

c) *ToString*

[C#]	public	int	Count	{get;}
[C++]	public:	__property	int	get_Count();
[VB]	Public	ReadOnly Property	Count	As Integer
[JScript]	public	function	get Count()	: int;

*Description*

Gets the number of items in the collection.

d) *IsReadOnly*

e) *ToString*

[C#]	public	bool	IsReadOnly	{get;}
[C++]	public:	__property	bool	get_IsReadOnly();
[VB]	Public	ReadOnly Property	IsReadOnly	As Boolean



[JScript]      public      function      get      IsReadOnly()      :      Boolean;

*Description*

Gets a value indicating if the collection is read-only.

*f)      Item*

*g)      ToString*

[C#]      public      object      this[int      index]      {get;      set;}

[C++]      public:      \_\_property      Object\*      get\_Item(int      index);public:      \_\_property      void  
set\_Item(int      index,      Object\*);

[VB]      Public      Default      Property      Item(ByVal      index      As      Integer)      As      Object

[JScript]      returnValue      =  
CheckedItemCollectionObject.Item(index);CheckedItemCollectionObject.Item(in  
dex)      =      returnValue;

*Description*

Gets an object in the checked items collection.

The **System.Windows.Forms.CheckedListBox.CheckedItems** collection is a subset of the objects in the **System.Windows.Forms.CheckedListBox.Items** collection, representing only items that are checked. This collection is ordered in ascending order. An index into the collection of checked items. This collection index corresponds to the index of the checked item.

*h)      Contains*

[C#]      public      bool      Contains(object      item);

[C++]      public:      \_\_sealed      bool      Contains(Object\*      item);

[VB] NotOverridable Public Function Contains(ByVal item As Object) As Boolean

[JScript] public function Contains(item : Object) : Boolean;

#### *Description*

Determines whether the specified item is located in the collection.

*Return Value:* **true** if item is in the collection; otherwise, **false**.

This method determines if an object from the **System.Windows.Forms.CheckedListBox.Items** collection is in the **System.Windows.Forms.CheckedListBox.CheckedItems** collection. An object from the items collection.

#### *i) CopyTo*

[C#] public void CopyTo(Array dest, int index);

[C++] public: \_\_sealed void CopyTo(Array\* dest, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)

[JScript] public function CopyTo(dest : Array, index : int);

#### *Description*

Copies the entire collection into an existing array at a specified location within the array.

You can use this method to combine the selected indexes from multiple collections into a single array. The destination array. The zero-based relative index in *dest* at which copying begins.

#### *j) GetEnumerator*

[C#] public IEnumerator GetEnumerator();

```

1  [C++]      public:      __sealed      IEnumerator*      GetEnumerator();
2  [VB] NotOverridable Public Function GetEnumerator() As IEnumerator
3  [JScript]  public      function      GetEnumerator()      :      IEnumerator;

```

#### *Description*

Returns an enumerator that can be used to iterate through the **System.Windows.Forms.CheckedListBox.CheckedItems** collection.  
*Return Value:* An **System.Collections.IEnumerator** for navigating through the list.

#### *k) IndexOf*

```

10 [C#]      public      int      IndexOf(object      item);
11
12 [C++]      public:      __sealed      int      IndexOf(Object*      item);
13
14 [VB] NotOverridable Public Function IndexOf(ByVal item As Object) As Integer
15
16 [JScript]  public      function      IndexOf(item      :      Object)      :      int;

```

#### *Description*

Returns an index into the collection of checked items.  
*Return Value:* The index of the object in the checked item collection or -1 if the object is not in the collection. For more information, see the examples in the **System.Windows.Forms.CheckedListBox.CheckedItemCollection** class overview.

The **System.Windows.Forms.CheckedListBox.CheckedItems** collection is a subset of the objects in the **System.Windows.Forms.CheckedListBox.Items** collection, representing only items that are checked. This collection is ordered in ascending order. The object whose index you want to retrieve. This object must belong to the checked items collection.

**l) *IList.Add***

[C#] int IList.Add(object value);  
[C++] int IList::Add(Object\* value);  
[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add  
[JScript] function IList.Add(value : Object) : int;

**m) *IList.Clear***

[C#] void IList.Clear();  
[C++] void IList::Clear();  
[VB] Sub Clear() Implements IList.Clear  
[JScript] function IList.Clear();

**n) *IList.Insert***

[C#] void IList.Insert(int index, object value);  
[C++] void IList::Insert(int index, Object\* value);  
[VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements IList.Insert  
[JScript] function IList.Insert(index : int, value : Object);

**o) *IList.Remove***

[C#] void IList.Remove(object value);  
[C++] void IList::Remove(Object\* value);  
[VB] Sub Remove(ByVal value As Object) Implements IList.Remove  
[JScript] function IList.Remove(value : Object);

p) *IList.RemoveAt*

```
[C#]          void          IList.RemoveAt(int          index);
[C++]          void          IList::RemoveAt(int          index);
[VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt
[JavaScript] function IList.RemoveAt(index : int);
```

CheckedListBox class (System.Windows.Forms)

a) *ToString*

*Description*

Displays a **System.Windows.Forms.ListBox** in which a check box is displayed to the left of each item.

This control presents a list of items that the user can navigate by using the keyboard or the scrollbar on the right side of the control. The user can place a check mark by one or more items and the checked items can be navigated with the **System.Windows.Forms.CheckedListBox.CheckedItemCollection** and **System.Windows.Forms.CheckedListBox.CheckedIndexCollection**.

b) *CheckedListBox*

*Example Syntax:*

c) *ToString*

```
[C#]          public          CheckedListBox();
[C++]          public:          CheckedListBox();
[VB]          Public          Sub          New()
[JavaScript]   public          function          CheckedListBox();
```

*Description*

Initializes a new instance of the **System.Windows.Forms.CheckedListBox** class.

By default, **System.Windows.Forms.CheckedListBox** uses **System.Windows.Forms.Control.SetStyle(System.Windows.Forms.ControlStyles, System.Boolean)** and **System.Windows.Forms.ControlStyles.ResizeRedraw** to specify that the control is redrawn when resized.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *BorderStyle*
- o) *Bottom*
- p) *Bounds*
- q) *CanFocus*
- r) *CanSelect*
- s) *Capture*
- t) *CausesValidation*
- u) *CheckedIndices*
- v) *ToString*

#### *Description*

Collection of checked indexes in this  
**System.Windows.Forms.CheckedListBox** .

The collection of checked indexes is a subset of the indexes into the collection of all items in the **System.Windows.Forms.CheckedListBox** control. These indexes specify items in a checked or indeterminate state.

w) *CheckedItems*

x) *ToString*

[C#] public CheckedListBox.CheckedItemCollection CheckedItems {get;}

[C++] public: \_\_property CheckedListBox.CheckedItemCollection\*  
get\_CheckedItems();

[VB] Public ReadOnly Property CheckedItems As  
CheckedListBox.CheckedItemCollection

[JScript] public function get CheckedItems() :  
CheckedListBox.CheckedItemCollection;

#### *Description*

Collection of checked items in this **System.Windows.Forms.CheckedListBox**.

The collection is a subset of the objects in the **System.Windows.Forms.CheckedListBox.Items** collection, representing only those items that are in a state of **System.Windows.Forms.CheckState.Checked** or **System.Windows.Forms.CheckState.Indeterminate**. The indexes in this collection are in ascending order.

y) *CheckOnClick*

z) *ToString*

[C#] public bool CheckOnClick {get; set;}

[C++] public: \_\_property bool get\_CheckOnClick();public: \_\_property void



1 set\_CheckOnClick(bool);

2 [VB]        Public        Property        CheckOnClick        As        Boolean

3 [JScript] public function get CheckOnClick() : Boolean;public function set

4 CheckOnClick(Boolean);

5  
6 *Description*

7 Gets or sets a value indicating whether the check box should be toggled when an  
8 item is selected.

9 **System.Windows.Forms.CheckedListBox.CheckOnClick** indicates whether  
10 the check box should be toggled whenever an item is selected. The default  
11 behavior is to change the selection on the first click, and then have the user click  
12 again to apply the check mark. In some instances, however, you might prefer  
13 have the item checked as soon as it is clicked.  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

- 1      **aa)    *ClientRectangle***
- 2      **bb)    *ClientSize***
- 3      **cc)    *ColumnWidth***
- 4      **dd)    *CompanyName***
- 5      **ee)    *Container***
- 6      **ff)    *ContainsFocus***
- 7      **gg)    *ContextMenu***
- 8      **hh)    *Controls***
- 9      **ii)    *Created***
- 10     **jj)    *CreateParams***
- 11     **kk)    *ToString***

#### *Description*

This property is read-only.

This is called when creating a window. Inheriting classes can override this to add extra functionality, but should not forget to first call `base.CreateParams()` to make sure the control continues to work correctly.

- 1) *Cursor*
- 2) *DataBindings*
- 3) *DataManager*
- 4) *DataSource*
- 5) *DefaultImeMode*
- 6) *DefaultSize*
- 7) *DesignMode*
- 8) *DisplayMember*
- 9) *DisplayRectangle*
- 10) *Disposing*
- 11) *Dock*
- 12) *DrawMode*
- 13) *ToString*

#### *Description*

Gets a value indicating the mode for drawing elements of the **System.Windows.Forms.CheckedListBox** .

yy)	<i>Enabled</i>
zz)	<i>Events</i>
aaa)	<i>Focused</i>
bbb)	<i>Font</i>
ccc)	<i>FontHeight</i>
ddd)	<i>ForeColor</i>
eee)	<i>Handle</i>
fff)	<i>HasChildren</i>
ggg)	<i>Height</i>
hhh)	<i>HorizontalExtent</i>
iii)	<i>HorizontalScrollbar</i>
jjj)	<i>ImeMode</i>
kkk)	<i>IntegralHeight</i>
lll)	<i>InvokeRequired</i>
mmm)	<i>IsAccessible</i>
nnn)	<i>IsDisposed</i>
ooo)	<i>IsHandleCreated</i>
ppp)	<i>ItemHeight</i>
qqq)	<i>ToString</i>

# *Description*

Gets the height of the item area.

This measurement is based on the font height plus a small margin to provide white space around the item.

*rrr) Items*

*sss) ToString*

[C#] public new CheckedListBox.ObjectCollection Items {get;}

[C++] public: \_\_property CheckedListBox.ObjectCollection\* get\_Items();

[VB] Public ReadOnly Property Items As CheckedListBox.ObjectCollection

[JScript] public function get Items() : CheckedListBox.ObjectCollection;

### *Description*

Gets the collection of items in this **System.Windows.Forms.CheckedListBox**

.

The **System.Windows.Forms.CheckedListBox.Items** property enables you to obtain a reference to the list of items that are currently stored in a

**System.Windows.Forms.CheckedListBox** control. With this reference, you can add items, remove items, and obtain a count of the items in the collection.

For more information on the tasks that can be performed with the item collection, see the

**System.Windows.Forms.CheckedListBox.ObjectCollection** class reference topics.

MS1-861US.APP  
509-324-9256  
lee@hayes.plc

1 **ttt) Left**

2 **uuu) Location**

3 **vvv) MultiColumn**

4 **www) Name**

5 **xxx) Parent**

6 **yyy) PreferredHeight**

7 **zzz) ProductName**

8 **aaaa) ProductVersion**

9 **bbbb) RecreatingHandle**

10 **cccc) Region**

11 **dddd) RenderRightToLeft**

12 **eeee) ResizeRedraw**

13 **ffff) Right**

14 **gggg) RightToLeft**

15 **hhhh) ScrollAlwaysVisible**

16 **iiii) SelectedIndex**

17 **jjjj) SelectedIndices**

18 **kkkk) SelectedItem**

19 **llll) SelectedItems**

20 **mmmm)SelectedValue**

21 **nnnn) SelectionMode**

22 **oooo) ToString**

*Description*

Gets or sets a value specifying the selection mode.

For **System.Windows.Forms.CheckedListBox** objects, multiple selection is not supported. You can set the mode to one item or no items.

*pppp) ShowFocusCues*

*qqqq) ShowKeyboardCues*

*rrrr) Site*

*ssss) Size*

*tttt) Sorted*

*uuuu) TabIndex*

*vvvv) TabStop*

*www)Tag*

*xxxx) Text*

*yyyy) ThreeDCheckBoxes*

*zzzz) ToString*

*Description*

Gets or sets a value indicating whether the check boxes display as **System.Windows.Forms.ButtonState.Flat** or **System.Windows.Forms.ButtonState.Normal** in appearance.

MSI-861US.APP

1	<i>aaaaa) Top</i>
2	<i>bbbbbb) TopIndex</i>
3	<i>ccccc) TopLevelControl</i>
4	<i>dddddd) UseTabStops</i>
5	<i>eeeeee) ValueMember</i>
6	<i>ffffff) Visible</i>
7	<i>gggggg) Width</i>
8	<i>hhhhh) WindowTarget</i>
9	<i>iiii) ToString</i>
10	
11	
12	<i>Description</i>
13	
14	<i>jjjjj) ToString</i>
15	
16	
17	<i>Description</i>
18	
19	<i>kkkkk) ToString</i>
20	
21	
22	<i>Description</i>
23	Occurs when the checked state of an item changes.
24	
25	



### IIII) ToString

#### Description

#### *mmmmm)CreateAccessibilityInstance*

```
[C#]    protected    override    AccessibleObject    CreateAccessibilityInstance();  
[C++]    protected:    AccessibleObject*    CreateAccessibilityInstance();  
[VB]    Overrides    Protected    Function    CreateAccessibilityInstance()    As  
AccessibleObject  
[JScript]    protected    override    function    CreateAccessibilityInstance()    :  
AccessibleObject;
```

#### Description

Constructs the new instance of the accessibility object for this control. Subclasses should not call base.CreateAccessibilityObject.

#### *nnnnn)CreateItemCollection*

```
[C#]    protected    override    ObjectCollection    CreateItemCollection();  
[C++]    protected:    ObjectCollection*    CreateItemCollection();  
[VB]    Overrides    Protected    Function    CreateItemCollection()    As    ObjectCollection  
[JScript]    protected    override    function    CreateItemCollection()    :    ObjectCollection;
```

#### Description

### ooooo) *GetItemChecked*

```
[C#]      public      bool      GetItemChecked(int      index);
[C++]      public:      bool      GetItemChecked(int      index);
[VB] Public Function GetItemChecked(ByVal index As Integer) As Boolean
[JScript] public function GetItemChecked(index : int) : Boolean;
```

#### *Description*

Returns a value indicating whether the specified item is checked.

*Return Value:* **true** if the item is checked; otherwise, **false** .

**System.Windows.Forms.CheckedListBox.GetItemChecked(System.Int32)** returns **true** if the item is **System.Windows.Forms.CheckState.Checked** or **System.Windows.Forms.CheckState.Indeterminate**. To determine the specific state the item is in, use the **System.Windows.Forms.CheckedListBox.GetItemCheckState(System.Int32)** method. The index of the item.

### ppppp) *GetItemCheckState*

```
[C#]      public      CheckState      GetItemCheckState(int      index);
[C++]      public:      CheckState      GetItemCheckState(int      index);
[VB] Public Function GetItemCheckState(ByVal index As Integer) As CheckState
[JScript] public function GetItemCheckState(index : int) : CheckState;
```

#### *Description*

Returns a value indicating the check state of the current item.

*Return Value:* One of the **System.Windows.Forms.CheckState** values. The index of the item to get the checked value of.

### **qqqqq) OnBackColorChanged**

[C#] protected override void OnBackColorChanged(EventArgs e);  
[C++] protected: void OnBackColorChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)  
[JScript] protected override function OnBackColorChanged(e : EventArgs);

#### *Description*

### **rrrrr) OnClick**

[C#] protected override void OnClick(EventArgs e);  
[C++] protected: void OnClick(EventArgs\* e);  
[VB] Overrides Protected Sub OnClick(ByVal e As EventArgs)  
[JScript] protected override function OnClick(e : EventArgs);

#### *Description*

Ensures that mouse clicks can toggle...

### **sssss) OnDrawItem**

[C#] protected override void OnDrawItem(DrawItemEventArgs e);  
[C++] protected: void OnDrawItem(DrawItemEventArgs\* e);  
[VB] Overrides Protected Sub OnDrawItem(ByVal e As DrawItemEventArgs)  
[JScript] protected override function OnDrawItem(e : DrawItemEventArgs);

## Description

Raises the DrawItem event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.Windows.Forms.DrawItemEventArgs** object with the details

### *ttttt) OnFontChanged*

[C#]      protected      override      void      OnFontChanged(EventArgs    e);

[C++]      protected:      void      OnFontChanged(EventArgs\*    e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs);

## Description

### *uuuuu)OnHandleCreated*

[C#]      protected      override      void      OnHandleCreated(EventArgs    e);

[C++]      protected:      void      OnHandleCreated(EventArgs\*    e);

[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

[JScript] protected override function OnHandleCreated(e : EventArgs);

## Description

When the handle is created we can dump any cached item-check pairs.

### *vvvvv) OnItemCheck*

```
[C#]    protected    virtual    void    OnItemCheck(ItemCheckEventArgs    ice);  
[C++]    protected:    virtual    void    OnItemCheck(ItemCheckEventArgs*    ice);  
[VB]    Overridable    Protected    Sub    OnItemCheck(ByVal    ice    As  
ItemCheckEventArgs)  
[JScript]    protected    function    OnItemCheck(ice    :    ItemCheckEventArgs);
```

#### *Description*

This is the code that actually fires the itemCheck event. Don't forget to call base.onItemCheck() to ensure that itemCheck vents are correctly fired for all other keys.

### *wwwww)OnKeyPress*

```
[C#]    protected    override    void    OnKeyPress(KeyPressEventArgs    e);  
[C++]    protected:    void    OnKeyPress(KeyPressEventArgs*    e);  
[VB]    Overrides    Protected    Sub    OnKeyPress(ByVal    e    As    KeyPressEventArgs)  
[JScript]    protected    override    function    OnKeyPress(e    :    KeyPressEventArgs);
```

#### *Description*

This is the code that actually fires the "keyPress" event. The Checked ListBox overrides this to look for space characters, since we want to use those to check or uncheck items periodically. Don't forget to call base.OnKeyPress() to ensure that KeyPrese events are correctly fired for all other keys. The KeyPressEventArgs that was fired.

### *xxxxx) OnMeasureItem*

```
[C#]    protected    override    void    OnMeasureItem(MeasureItemEventArgs    e);
```

1 [C++] protected: void OnMeasureItem(MeasureItemEventArgs\* e);

2 [VB] Overrides Protected Sub OnMeasureItem(ByVal e As  
3 MeasureItemEventArgs)

4 [JScript] protected override function OnMeasureItem(e : MeasureItemEventArgs);

5  
6 *Description*

7 Raises the MeasureItem event.

8 Raising an event invokes the event handler through a delegate. For more  
9 information, see .

10 *yyyyy) OnSelectedIndexChanged*

11 [C#] protected override void OnSelectedIndexChanged(EventArgs e);

12 [C++] protected: void OnSelectedIndexChanged(EventArgs\* e);

13 [VB] Overrides Protected Sub OnSelectedIndexChanged(ByVal e As EventArgs)

14 [JScript] protected override function OnSelectedIndexChanged(e : EventArgs);

15  
16 *Description*

17 Actually goes and fires the selectedIndexChanged event. Inheriting controls  
18 should use this to know when the event is fired [this is preferable to adding an  
19 event handler on yourself for this event]. They should, however, remember to  
20 call base.OnSelectedIndexChanged(e); to ensure the event is still fired to  
21 external listeners Actually goes and fires the selectedIndexChanged event.  
22 Inheriting controls should use this to know when the event is fired [this is  
23 preferable to adding an event handler on yourself for this event]. They should,  
24 however, remember to call base.OnSelectedIndexChanged(e); to ensure the  
25 event is still fired to external listeners Event object with the details

23 *zzzzz) SetItemChecked*

24  
25 [C#] public void SetItemChecked(int index, bool value);

1 [C++] public: void SetItemChecked(int index, bool value);

2 [VB] Public Sub SetItemChecked(ByVal index As Integer, ByVal value As  
3 Boolean)

4 [JScript] public function SetItemChecked(index : int, value : Boolean);

5  
6 *Description*

7 Sets the item at the specified index to  
8 **System.Windows.Forms.CheckState.Checked** .

9 When a value of **true** is passed, this method sets the checked value of the item  
10 to **System.Windows.Forms.CheckState.Checked** . A value of **false** will set  
11 the item to **System.Windows.Forms.CheckState.Unchecked** . The index of  
12 the item to set the check state for. **true** to set the item as checked; otherwise,  
13 **false**.

14 *aaaaaa)SetItemCheckState*

15 [C#] public void SetItemCheckState(int index, CheckState value);

16 [C++] public: void SetItemCheckState(int index, CheckState value);

17 [VB] Public Sub SetItemCheckState(ByVal index As Integer, ByVal value As  
18 CheckState)

19 [JScript] public function SetItemCheckState(index : int, value : CheckState);

20 *Description*

21 Sets the check state of the item at the specified index.

22 The  
23 **System.Windows.Forms.CheckedListBox.SetItemCheckState(System.In  
24 t32,System.Windows.Forms.CheckState)** method raises the  
25 **System.Windows.Forms.CheckedListBox.ItemCheck** event. The index of  
the item to set the state for. One of the **System.Windows.Forms.CheckState**  
values.

### ***bbbbbb)WmReflectCommand***

```
[C#] protected override void WmReflectCommand(ref Message m);
[C++] protected: void WmReflectCommand(Message* m);
[VB] Overrides Protected Sub WmReflectCommand(ByRef m As Message)
[JScript] protected override function WmReflectCommand(m : Message);
```

#### ***Description***

We need to get LBN\_SELCHANGE notifications We need to get LBN\_SELCHANGE notifications

### ***cccccc)WndProc***

```
[C#] protected override void WndProc(ref Message m);
[C++] protected: void WndProc(Message* m);
[VB] Overrides Protected Sub WndProc(ByRef m As Message)
[JScript] protected override function WndProc(m : Message);
```

#### ***Description***

The listbox's window procedure. Inheriting classes can override this to add extra functionality, but should not forget to call base.wndProc(m); to ensure the button continues to function properly. A Windows Message Object.

ListView.CheckedListViewItemCollection class (System.Windows.Forms)

#### ***a) WndProc***

#### ***Description***



**b) *ListView.CheckedListViewItemCollection***

*Example Syntax:*

**c) *WndProc***

```
[C#]    public    ListView.CheckedListViewItemCollection(ListView    owner);
[C++]    public:    CheckedListViewItemCollection(ListView*    owner);
[VB]    Public    Sub    New(ByVal    owner    As    ListView)
[JScript]    public    function    ListView.CheckedListViewItemCollection(owner    :
ListView);
```

*Description*

**d) *Count***

**e) *WndProc***

```
[C#]                public                int                Count                {get;}
[C++]                public:                __property                int                get_Count();
[VB]    Public    ReadOnly    Property    Count    As    Integer
[JScript]    public    function    get    Count()    :    int;
```

*Description*

Gets the number of currently selected items.

*f) IsReadOnly*

*g) WndProc*

```
[C#]      public      bool      IsReadOnly      {get;}
[C++]      public:      __property      bool      get_IsReadOnly();
[VB]      Public      ReadOnly      Property      IsReadOnly      As      Boolean
[JScript]      public      function      get      IsReadOnly()      :      Boolean;
```

*Description*

Gets whether the collection is read-only.

*h) Item*

*i) WndProc*

```
[C#]      public      ListViewItem      this[int      index]      {get;}
[C++]      public:      __property      ListViewItem*      get_Item(int      index);
[VB]      Public      Default      ReadOnly      Property      Item(ByVal      index      As      Integer)      As
ListViewItem
[JScript]      returnValue      =      CheckedListViewItemCollectionObject.Item(index);
```

*Description*

Selected item in the list.

*j) Contains*

```
[C#]      public      bool      Contains(ListViewItem      item);
[C++]      public:      bool      Contains(ListViewItem*      item);
```

1 [VB] Public Function Contains(ByVal item As ListViewItem) As Boolean

2 [JScript] public function Contains(item : ListViewItem) : Boolean;

3  
4 *Description*

5  
6 **k) CopyTo**

7  
8 [C#] public void CopyTo(Array dest, int index);

9 [C++] public: \_\_sealed void CopyTo(Array\* dest, int index);

10 [VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As  
11 Integer)

12 [JScript] public function CopyTo(dest : Array, index : int);

13  
14 *Description*

15  
16 **l) GetEnumerator**

17  
18 [C#] public IEnumerator GetEnumerator();

19 [C++] public: \_\_sealed IEnumerator\* GetEnumerator();

20 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

21 [JScript] public function GetEnumerator() : IEnumerator;

22  
23 *Description*

24

25

**m) IndexOf**

[C#] public int IndexOf(ListViewItem item);  
[C++] public: int IndexOf(ListViewItem\* item);  
[VB] Public Function IndexOf(ByVal item As ListViewItem) As Integer  
[JScript] public function IndexOf(item : ListViewItem) : int;

*Description*

**n) IList.Add**

[C#] int IList.Add(object value);  
[C++] int IList::Add(Object\* value);  
[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add  
[JScript] function IList.Add(value : Object) : int;

**o) IList.Clear**

[C#] void IList.Clear();  
[C++] void IList::Clear();  
[VB] Sub Clear() Implements IList.Clear  
[JScript] function IList.Clear();

**p) IList.Contains**

[C#] bool IList.Contains(object item);  
[C++] bool IList::Contains(Object\* item);

1 [VB] Function Contains(ByVal item As Object) As Boolean Implements

2 IList.Contains

3 [JScript] function IList.Contains(item : Object) : Boolean;

4 **q) *IList.IndexOf***

6 [C#] int IList.IndexOf(object item);

7 [C++] int IList::IndexOf(Object\* item);

8 [VB] Function IndexOf(ByVal item As Object) As Integer Implements

9 IList.IndexOf

10 [JScript] function IList.IndexOf(item : Object) : int;

11 **r) *IList.Insert***

13 [C#] void IList.Insert(int index, object value);

14 [C++] void IList::Insert(int index, Object\* value);

15 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

16 IList.Insert

17 [JScript] function IList.Insert(index : int, value : Object);

18 **s) *IList.Remove***

20 [C#] void IList.Remove(object value);

21 [C++] void IList::Remove(Object\* value);

22 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

23 [JScript] function IList.Remove(value : Object);

t) *IList.RemoveAt*

[C#] void IList.RemoveAt(int index);  
 [C++] void IList::RemoveAt(int index);  
 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt  
 [JScript] function IList.RemoveAt(index : int);  
 CheckState enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies the state of a control, such as a check box, that can be checked, unchecked, or set to an indeterminate state.

Multiple methods in **System.Windows.Forms.CheckBox** , **System.Windows.Forms.CheckedListBox** , and **System.Windows.Forms.ItemCheckEventArgs** use this enumeration.

b) *ToString*

[C#] public const CheckState Checked;  
 [C++] public: const CheckState Checked;  
 [VB] Public Const Checked As CheckState  
 [JScript] public var Checked : CheckState;

*Description*

The control is checked.

**c) ToString**

[C#]	public	const	CheckState	Indeterminate;
[C++]	public:	const	CheckState	Indeterminate;
[VB]	Public	Const	Indeterminate	As CheckState
[JScript]	public	var	Indeterminate	: CheckState;

**Description**

The control is indeterminate. An indeterminate control generally has a shaded appearance.

**d) ToString**

[C#]	public	const	CheckState	Unchecked;
[C++]	public:	const	CheckState	Unchecked;
[VB]	Public	Const	Unchecked	As CheckState
[JScript]	public	var	Unchecked	: CheckState;

**Description**

The control is unchecked.

ComboBox.ChildAccessibleObject class (System.Windows.Forms)

**a) ToString**

**Description**

**b) *ComboBox.ChildAccessibleObject***

*Example Syntax:*

**c) *ToString***

```
[C#] public ComboBox.ChildAccessibleObject(ComboBox owner, IntPtr handle);  
[C++] public: ChildAccessibleObject(ComboBox* owner, IntPtr handle);  
[VB] Public Sub New(ByVal owner As ComboBox, ByVal handle As IntPtr)  
[JScript] public function ComboBox.ChildAccessibleObject(owner : ComboBox,  
handle                                     :                               IntPtr);
```

*Description*

**d) *Bounds***

**e) *DefaultAction***

**f) *Description***

**g) *Help***

**h) *KeyboardShortcut***

**i) *Name***

**j) *ToString***

*Description*



1        *k)      Parent*

2        *l)      Role*

3        *m)      State*

4        *n)      Value*

5        Clipboard class (System.Windows.Forms)

6        *a)      UseStdAccessibleObjects*

7  
8  
9        *Description*

10       Provides methods to place data on and retrieve data from the system Clipboard.  
11       This class cannot be inherited.

12       For a list of predefined formats to use with the  
13       **System.Windows.Forms.Clipboard** class, see the  
14       **System.Windows.Forms.DataFormats** class.

15       *b)      GetDataObject*

16       [C#]        public        static        IDataObject        GetDataObject();

17       [C++]        public:        static        IDataObject\*        GetDataObject();

18       [VB]        Public        Shared        Function        GetDataObject()        As        IDataObject

19       [JScript]    public        static        function        GetDataObject()        :        IDataObject;

20  
21       *Description*

22       Retrieves the data that is currently on the system Clipboard.

23       *Return Value:* An **System.Windows.Forms.IDataObject** that represents the  
24       data currently on the Clipboard, or **null** if there is no data on the Clipboard.

25       Because the data type of the object returned from the Clipboard can vary, this  
26       method returns the data in an **System.Windows.Forms.IDataObject** . Then

you can use methods of the **System.Windows.Forms.IDataObject** interface to extract the data in its proper data type.

*c) SetDataObject*

[C#] public static void SetDataObject(object data);

[C++] public: static void SetDataObject(Object\* data);

[VB] Public Shared Sub SetDataObject(ByVal data As Object)

[JScript] public static function SetDataObject(data : Object); Places data on the system Clipboard.

*Description*

Places nonpersistent data on the system Clipboard.

Data will be deleted from system Clipboard when the application exits. The data to place on the Clipboard.

*d) SetDataObject*

[C#] public static void SetDataObject(object data, bool copy);

[C++] public: static void SetDataObject(Object\* data, bool copy);

[VB] Public Shared Sub SetDataObject(ByVal data As Object, ByVal copy As Boolean)

[JScript] public static function SetDataObject(data : Object, copy : Boolean);

*Description*

Places data on the system Clipboard and specifies whether the data should remain on the Clipboard after the application exits.

If the *copy* parameter is **false** , the data will be deleted from system Clipboard when the application exits. The data to place on the Clipboard. **true** if you want data to remain on the Clipboard after this application exits; otherwise, **false**.

AxHost.ClsidAttribute class (System.Windows.Forms)

- a) ToString
- b) AxHost.ClsidAttribute

Example Syntax:

- c) ToString
- d) TypeId
- e) Value
- f) ToString

ColorDepth enumeration (System.Windows.Forms)

- a) ToString

Description

Specifies the number of colors used to display an image in an **System.Windows.Forms.ImageList** control.

This enumeration is used by the **System.Windows.Forms.ImageList.ColorDepth** property of the **System.Windows.Forms.ImageList** class.

- b) ToString

[C#]	public	const	ColorDepth	Depth16Bit;
[C++]	public:	const	ColorDepth	Depth16Bit;
[VB]	Public	Const	Depth16Bit	As ColorDepth

[JScript]      public      var      Depth16Bit      :      ColorDepth;

*Description*

A 16-bit image.

*c) ToString*

[C#]      public      const      ColorDepth      Depth24Bit;

[C++]      public:      const      ColorDepth      Depth24Bit;

[VB]      Public      Const      Depth24Bit      As      ColorDepth

[JScript]      public      var      Depth24Bit      :      ColorDepth;

*Description*

A 24-bit image.

*d) ToString*

[C#]      public      const      ColorDepth      Depth32Bit;

[C++]      public:      const      ColorDepth      Depth32Bit;

[VB]      Public      Const      Depth32Bit      As      ColorDepth

[JScript]      public      var      Depth32Bit      :      ColorDepth;

*Description*

A 32-bit image.

*e) ToString*

[C#]      public      const      ColorDepth      Depth4Bit;

[C++]	public:	const	ColorDepth	Depth4Bit;
[VB]	Public	Const	Depth4Bit	As ColorDepth
[JScript]	public	var	Depth4Bit	: ColorDepth;

*Description*

A 4-bit image.

*f) ToString*

[C#]	public	const	ColorDepth	Depth8Bit;
[C++]	public:	const	ColorDepth	Depth8Bit;
[VB]	Public	Const	Depth8Bit	As ColorDepth
[JScript]	public	var	Depth8Bit	: ColorDepth;

*Description*

An 8-bit image.

ColorDialog class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a common dialog box that displays available colors along with controls that allow the user to define custom colors.

The inherited member

**System.Windows.Forms.CommonDialog.ShowDialog** must be invoked to create this specific common dialog box.

**b) ColorDialog**

*Example Syntax:*

**c) ToString**

[C#]	public	ColorDialog();
[C++]	public:	ColorDialog();
[VB]	Public Sub	New()
[JScript]	public function	ColorDialog();

*Description*

Initializes a new instance of the **System.Windows.Forms.ColorDialog** class.

When you create an instance of **System.Windows.Forms.ColorDialog**, the following read/write properties are set to initial values.

**d) AllowFullOpen**

**e) ToString**

[C#]	public	virtual	bool	AllowFullOpen	{get; set;}
[C++]	public: __property	virtual	bool	get_AllowFullOpen();	public: __property
	virtual		void	set_AllowFullOpen(bool);	
[VB]	Overridable	Public	Property	AllowFullOpen	As Boolean
[JScript]	public function	get	AllowFullOpen()	: Boolean;	public function
			set	AllowFullOpen(Boolean);	

*Description*

Gets or sets a value indicating whether the user can use the dialog box to define custom colors.

When set to **false** , the associated button in the dialog box is disabled and the user cannot access the custom colors control in the dialog box.

*f) AnyColor*

*g) ToString*

[C#] public virtual bool AnyColor {get; set;}

[C++] public: \_\_property virtual bool get\_AnyColor();public: \_\_property virtual void set\_AnyColor(bool);

[VB] Overridable Public Property AnyColor As Boolean

[JScript] public function get AnyColor() : Boolean;public function set AnyColor(Boolean);

#### *Description*

Gets or sets a value indicating whether the dialog box displays all available colors in the set of basic colors.

*h) Color*

*i) ToString*

[C#] public Color Color {get; set;}

[C++] public: \_\_property Color get\_Color();public: \_\_property void set\_Color(Color);

[VB] Public Property Color As Color

[JScript] public function get Color() : Color;public function set Color(Color);

#### *Description*

Gets or sets the color selected by the user.

The color selected by the user in the dialog box at run time, as defined in **System.Drawing.Color** structure.

*j) Container*

*k) CustomColors*

*l) ToString*

#### *Description*

Gets or sets the set of custom colors shown in the dialog box.

Users can create their own set of custom colors.

*m) DesignMode*

*n) Events*

*o) FullOpen*

*p) ToString*

#### *Description*

Gets or sets a value indicating whether the controls used to create custom colors are visible when the dialog box is opened **true** if the custom color controls are available when the dialog box is opened; otherwise, **false** . The default value is **false** .

By default, the custom color controls are not visible when the dialog box is first opened. You must click the Custom Colors button to display them.



1           **q)     Instance**

2           **r)     ToString**

3  
4 [C#]       protected       virtual       IntPtr       Instance       {get;}

5 [C++]      protected:   \_\_property   virtual   IntPtr    get\_Instance();

6 [VB]   Overridable   Protected   ReadOnly   Property   Instance   As   IntPtr

7 [JScript]   protected   function   get    Instance()   :   IntPtr;

8  
9 *Description*

10 Our HINSTANCE from Windows.

11           **s)     Options**

12           **t)     ToString**

13  
14 [C#]       protected       virtual       int       Options       {get;}

15 [C++]      protected:   \_\_property   virtual   int    get\_Options();

16 [VB]   Overridable   Protected   ReadOnly   Property   Options   As   Integer

17 [JScript]   protected   function   get    Options()   :   int;

18  
19 *Description*

20 Returns our CHOOSECOLOR options.

21           **u)     ShowHelp**

22           **v)     ToString**

23  
24 [C#]       public       virtual       bool       ShowHelp       {get;    set;}

25 [C++]      public:   \_\_property   virtual   bool   get\_ShowHelp();public:   \_\_property   virtual

```

1 void set_ShowHelp(bool);
2 [VB] Overridable Public Property ShowHelp As Boolean
3 [JScript] public function get ShowHelp() : Boolean;public function set
4 ShowHelp(Boolean);
5

```

#### *Description*

Gets or sets a value indicating whether a Help button appears in the color dialog box.

w) *Site*

x) *SolidColorOnly*

y) *ToString*

#### *Description*

Gets or sets a value indicating whether the dialog box will restrict users to selecting solid colors only.

This property is applicable to systems with 256 or fewer colors. On these types of systems, some colors are composites of others.

z) *Reset*

[C#]	public	override	void	Reset();
[C++]	public:		void	Reset();
[VB]	Overrides	Public	Sub	Reset()
[JScript]	public	override	function	Reset();

#### *Description*

Resets all options to their default values, the last selected color to black, and the custom colors to their default values.

**aa) RunDialog**

[C#] protected override bool RunDialog(IntPtr hwndOwner);

[C++] protected: bool RunDialog(IntPtr hwndOwner);

[VB] Overrides Protected Function RunDialog(ByVal hwndOwner As IntPtr) As Boolean

[JScript] protected override function RunDialog(hwndOwner : IntPtr) : Boolean;

*Description*

**bb) ToString**

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

*Description*

Provides a string version of this object.

*Return Value:* A string version of this object.

ColumnClickEventArgs class (System.Windows.Forms)

**a) ToString**

*Description*

Provides data for the **System.Windows.Forms.ListView.ColumnClick** event.

The **System.Windows.Forms.ColumnClickEventArgs** class provides the zero-based index within the **System.Windows.Forms.ListView.ColumnHeaderCollection** class of the column that is clicked in the **System.Windows.Forms.ListView** control. You can use this information in an event handler for the **System.Windows.Forms.ListView.ColumnClick** event to determine which column is being clicked to perform tasks on the data within the column.

*b) ColumnClickEventArgs*

*Example Syntax:*

*c) ToString*

[C#]	public	ColumnClickEventArgs(int	column);
[C++]	public:	ColumnClickEventArgs(int	column);
[VB]	Public	Sub New(ByVal column	As Integer)
[JScript]	public	function ColumnClickEventArgs(column	: int);

*Description*

Initializes a new instance of the **System.Windows.Forms.ColumnClickEventArgs** class. The zero-based index of the column that is clicked.

*d) Column*

*e) ToString*

[C#]	public	int	Column	{get;}
[C++]	public:	__property	int	get_Column();
[VB]	Public	ReadOnly	Property	Column As Integer
[JScript]	public	function	get Column()	: int;

## Description

Gets the zero-based index of the column that is clicked.

You can use the information provided by this property in an event handler for the **System.Windows.Forms.ListView.ColumnClick** event to determine which column is being clicked to perform tasks on the data within the column.

ColumnClickEventHandler delegate (System.Windows.Forms)

### a) ToString

## Description

Represents the method that will handle the the **System.Windows.Forms.ListView.ColumnClick** event of a **System.Windows.Forms.ListView** . The source of the event. A **System.Windows.Forms.ColumnClickEventArgs** that contains the event data.

When you create a **System.Windows.Forms.ColumnClickEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ColumnHeader class (System.Windows.Forms)

### a) ToString

## Description

Displays a single column header in a **System.Windows.Forms.ListView** control.

A column header is an item in a **System.Windows.Forms.ListView** control that contains heading text.

**b) ColumnHeader**

*Example Syntax:*

**c) ToString**

[C#]	public	ColumnHeader();
[C++]	public:	ColumnHeader();
[VB]	Public	Sub New()
[JScript]	public	function ColumnHeader();

*Description*

Initializes a new instance of the **System.Windows.Forms.ColumnHeader** class.

**d) Container**

**e) DesignMode**

**f) Events**

**g) Index**

**h) ToString**

*Description*

Gets the location with the **System.Windows.Forms.ListView** control's **System.Windows.Forms.ListView.ColumnHeaderCollection** of this column.

The value of this property does not necessarily correspond to the current visual position of the column header within the **System.Windows.Forms.ListView** . This can be due to the user reordering the column headers at run time (when the **System.Windows.Forms.ListView.AllowColumnReorder** property is set to **true** ). If the **System.Windows.Forms.ColumnHeader** is not contained

within a **System.Windows.Forms.ListView** control this property returns a value of -1.

i) *ListView*

j) *ToString*

```
[C#]          public          ListView          ListView          {get;}
```

```
[C++]         public:         __property         ListView*         get_ListView();
```

```
[VB]          Public          ReadOnly          Property          ListView          As          ListView
```

```
[JScript]     public          function          get          ListView()          :          ListView;
```

#### *Description*

Gets the **System.Windows.Forms.ListView** control the **System.Windows.Forms.ColumnHeader** is located in.

You can use this property to determine which **System.Windows.Forms.ListView** control a specific **System.Windows.Forms.ColumnHeader** object is associated with.

k) *Site*

l) *Text*

m) *ToString*

#### *Description*

Gets or sets the text displayed in the column header.

n) *TextAlign*

o) *ToString*

```
[C#]      public      HorizontalAlignment      TextAlign      {get;      set;}
[C++]      public:      __property      HorizontalAlignment      get_TextAlign();public:
__property      void      set_TextAlign(HorizontalAlignment);
[VB]      Public      Property      TextAlign      As      HorizontalAlignment
[JScript] public function get TextAlign() : HorizontalAlignment;public function
set      TextAlign(HorizontalAlignment);
```

#### *Description*

Gets or sets the horizontal alignment of the text displayed in the **System.Windows.Forms.ColumnHeader**.

You can use this property to provide different text alignment settings for the text displayed in each **System.Windows.Forms.ColumnHeader**. For example, you might want to align the first **System.Windows.Forms.ColumnHeader** in a **System.Windows.Forms.ListView** control to the left while keeping the rest of the fields center or right aligned.

p) *Width*

q) *ToString*

```
[C#]      public      int      Width      {get;      set;}
[C++]      public:      __property      int      get_Width();public:      __property      void      set_Width(int);
[VB]      Public      Property      Width      As      Integer
[JScript] public function get Width() : int;public function set Width(int);
```

#### *Description*



Gets or sets the width of the column.

*r) Clone*

[C#]	public	object	Clone();
[C++]	public:	__sealed	Object* Clone();
[VB]	NotOverridable	Public	Function Clone() As Object
[JScript]	public	function	Clone() : Object;

*Description*

Creates an identical copy of the current **System.Windows.Forms.ColumnHeader** that is not attached to any list view control.

*Return Value:* An object representing a copy of this **System.Windows.Forms.ColumnHeader** object.

You can use this method to copy existing **System.Windows.Forms.ColumnHeader** objects from an existing **System.Windows.Forms.ListView** control for use in other **System.Windows.Forms.ListView** controls.

*s) Dispose*

[C#]	protected	override	void	Dispose(bool disposing);
[C++]	protected:		void	Dispose(bool disposing);
[VB]	Overrides	Protected	Sub	Dispose(ByVal disposing As Boolean)
[JScript]	protected	override	function	Dispose(disposing : Boolean);

*Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.ColumnHeader**.

Call **System.Windows.Forms.ColumnHeader.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.ColumnHeader** . The **System.Windows.Forms.ColumnHeader.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.ColumnHeader** in an unusable state. After calling **System.Windows.Forms.ColumnHeader.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.ColumnHeader** so the memory it was occupying can be reclaimed by garbage collection.

*t) ToString*

[C#]	public	override	string	ToString();
[C++]	public:		String*	ToString();
[VB]	Overrides	Public	Function	ToString() As String
[JScript]	public	override	function	ToString() : String;

*Description*

Returns a string representation of this column header Returns a string representation of this column header

ListView.ColumnHeaderCollection class (System.Windows.Forms)

*a) ToString*

*Description*

*b) ListView.ColumnHeaderCollection*

*Example Syntax:*

c) *ToString*

```
[C#]      public      ListView.ColumnHeaderCollection(ListView      owner);
[C++]      public:      ColumnHeaderCollection(ListView*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      ListView)
[JScript] public function ListView.ColumnHeaderCollection(owner : ListView);
```

*Description*

d) *Count*

e) *ToString*

```
[C#]      public      int      Count      {get;}
[C++]      public:      __property      int      get_Count();
[VB]      Public      ReadOnly      Property      Count      As      Integer
[JScript] public      function      get      Count()      :      int;
```

*Description*

Gets the number of columns the **System.Windows.Forms.ListView** currently has in Details view.

f) *IsReadOnly*

g) *ToString*

```
[C#]      public      bool      IsReadOnly      {get;}
[C++]      public:      __property      bool      get_IsReadOnly();
```

1 [VB] Public ReadOnly Property IsReadOnly As Boolean

2 [JScript] public function get IsReadOnly() : Boolean;

3  
4 *Description*

5  
6 *h) Item*

7 *i) ToString*

8  
9 [C#] public virtual ColumnHeader this[int index] {get;}

10 [C++] public: \_\_property virtual ColumnHeader\* get\_Item(int index);

11 [VB] Overridable Public Default ReadOnly Property Item(ByVal index As  
12 Integer) As ColumnHeader

13 [JScript] returnValue = ColumnHeaderCollectionObject.Item(index);

14  
15 *Description*

16 Given a zero-based index, returns the  
17 **System.Windows.Forms.ColumnHeader** object for the column at that index.  
The zero-based index of specific column.

18 *j) Add*

19  
20 [C#] public virtual int Add(ColumnHeader value);

21 [C++] public: virtual int Add(ColumnHeader\* value);

22 [VB] Overridable Public Function Add(ByVal value As ColumnHeader) As  
23 Integer

24 [JScript] public function Add(value : ColumnHeader) : int;

1  
2 *Description*

3  
4 *k) Add*

5  
6 [C#] public virtual ColumnHeader Add(string str, int width, HorizontalAlignment  
7 textAlign);

8 [C++] public: virtual ColumnHeader\* Add(String\* str, int width,  
9 HorizontalAlignment textAlign);

10 [VB] Overridable Public Function Add(ByVal str As String, ByVal width As  
11 Integer, ByVal textAlign As HorizontalAlignment) As ColumnHeader

12 [JScript] public function Add(str : String, width : int, textAlign :  
13 HorizontalAlignment) : ColumnHeader;

14  
15 *Description*

16 Adds a column to the end of the list.

17 *Return Value:* A **System.Windows.Forms.ColumnHeader** object representing  
18 the new column. Text for new column. Width of new column in pixels. The  
19 alignment of the text in this column. Must be a value from  
20 **System.Windows.Forms.HorizontalAlignment** class.

21  
22 *l) AddRange*

23 [C#] public virtual void AddRange(ColumnHeader[] values);

24 [C++] public: virtual void AddRange(ColumnHeader\* values[]);

25 [VB] Overridable Public Sub AddRange(ByVal values() As ColumnHeader)

[JScript] public function AddRange(values : ColumnHeader[]);

## Description

### *m) Clear*

[C#]	public	virtual	void	Clear();
[C++]	public:	virtual	void	Clear();
[VB]	Overridable	Public	Sub	Clear()
[JScript]	public	function		Clear();

## Description

Removes all columns from the list view.

### *n) Contains*

[C#]	public	bool	Contains(ColumnHeader	value);
[C++]	public:	bool	Contains(ColumnHeader*	value);
[VB]	Public Function	Contains(ByVal value As ColumnHeader)	As Boolean	
[JScript]	public	function	Contains(value : ColumnHeader)	: Boolean;

## Description

### *o) GetEnumerator*

[C#]	public	IEnumerator	GetEnumerator();	
[C++]	public:	__sealed	IEnumerator*	GetEnumerator();

1 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

2 [JScript] public function GetEnumerator() : IEnumerator;

3  
4 *Description*

5  
6 **p) IndexOf**

7  
8 [C#] public int IndexOf(ColumnHeader value);

9 [C++] public: int IndexOf(ColumnHeader\* value);

10 [VB] Public Function IndexOf(ByVal value As ColumnHeader) As Integer

11 [JScript] public function IndexOf(value : ColumnHeader) : int;

12  
13 *Description*

14  
15 **q) Insert**

16  
17 [C#] public void Insert(int index, ColumnHeader value);

18 [C++] public: void Insert(int index, ColumnHeader\* value);

19 [VB] Public Sub Insert(ByVal index As Integer, ByVal value As ColumnHeader)

20 [JScript] public function Insert(index : int, value : ColumnHeader);

21  
22 *Description*

**r) Insert**

[C#] public void Insert(int index, string str, int width, HorizontalAlignment  
textAlign);

[C++] public: void Insert(int index, String\* str, int width, HorizontalAlignment  
textAlign);

[VB] Public Sub Insert(ByVal index As Integer, ByVal str As String, ByVal width  
As Integer, ByVal textAlign As HorizontalAlignment)

[JScript] public function Insert(index : int, str : String, width : int, textAlign :  
HorizontalAlignment);

*Description*

**s) Remove**

[C#] public virtual void Remove(ColumnHeader column);

[C++] public: virtual void Remove(ColumnHeader\* column);

[VB] Overridable Public Sub Remove(ByVal column As ColumnHeader)

[JScript] public function Remove(column : ColumnHeader);

*Description*

**t) RemoveAt**

[C#] public virtual void RemoveAt(int index);



```

1 [C++]      public:      virtual      void      RemoveAt(int      index);
2 [VB]  Overridable  Public  Sub  RemoveAt(ByVal  index  As  Integer)
3 [JScript]      public      function      RemoveAt(index      :      int);

```

#### 5 *Description*

6 Removes a column from the **System.Windows.Forms.ListView** . The zero-based index of the column that is to be removed

#### 7 *u) ICollection.CopyTo*

```

8
9 [C#]      void      ICollection.CopyTo(Array      dest,      int      index);
10 [C++]      void      ICollection::CopyTo(Array*      dest,      int      index);
11 [VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
12 ICollection.CopyTo
13 [JScript] function ICollection.CopyTo(dest : Array, index : int);

```

#### 14 *v) IList.Add*

```

15
16 [C#]      int      IList.Add(object      value);
17 [C++]      int      IList::Add(Object*      value);
18 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add
19 [JScript] function IList.Add(value : Object) : int;

```

#### 20 *w) IList.Contains*

```

21
22 [C#]      bool      IList.Contains(object      value);
23 [C++]      bool      IList::Contains(Object*      value);
24 [VB] Function Contains(ByVal value As Object) As Boolean Implements
25

```

1 IList.Contains

2 [JScript] function IList.Contains(value : Object) : Boolean;

3 **x) IList.IndexOf**

4  
5 [C#] int IList.IndexOf(object value);

6 [C++] int IList::IndexOf(Object\* value);

7 [VB] Function IndexOf(ByVal value As Object) As Integer Implements

8 IList.IndexOf

9 [JScript] function IList.IndexOf(value : Object) : int;

10 **y) IList.Insert**

11  
12 [C#] void IList.Insert(int index, object value);

13 [C++] void IList::Insert(int index, Object\* value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17 **z) IList.Remove**

18  
19 [C#] void IList.Remove(object value);

20 [C++] void IList::Remove(Object\* value);

21 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

22 [JScript] function IList.Remove(value : Object);

ColumnHeaderStyle enumeration (System.Windows.Forms)

a) ToString

Description

Specifies the styles of the column headers in a **System.Windows.Forms.ListView** control.

Use the members of this enumeration to set the value of the **System.Windows.Forms.ListView.HeaderStyle** property of the **System.Windows.Forms.ListView** control.

b) ToString

[C#]	public	const	ColumnHeaderStyle	Clickable;
[C++]	public:	const	ColumnHeaderStyle	Clickable;
[VB]	Public	Const	Clickable	As ColumnHeaderStyle
[JScript]	public	var	Clickable	: ColumnHeaderStyle;

Description

The column headers function like buttons and can carry out an action, such as sorting, when clicked.

c) ToString

[C#]	public	const	ColumnHeaderStyle	Nonclickable;
[C++]	public:	const	ColumnHeaderStyle	Nonclickable;
[VB]	Public	Const	Nonclickable	As ColumnHeaderStyle
[JScript]	public	var	Nonclickable	: ColumnHeaderStyle;

## Description

The column headers do not respond to the click of a mouse.

### d) ToString

[C#]	public	const	ColumnHeaderStyle	None;
[C++]	public:	const	ColumnHeaderStyle	None;
[VB]	Public	Const	None	As ColumnHeaderStyle
[JScript]	public	var	None	: ColumnHeaderStyle;

## Description

The column header is not displayed in report view.

ComboBox class (System.Windows.Forms)

### a) ToString

## Description

Represents a Windows combo box control.

A **System.Windows.Forms.ComboBox** displays an editing field combined with a **System.Windows.Forms.ListBox**, allowing the user to select from the list or to enter new text. The default behaviour of **System.Windows.Forms.ComboBox** displays an edit field with a hidden drop-down list. The **System.Windows.Forms.ComboBox.DropDownStyle** property determines the style of combo box to display. You can enter a value that allows for a simple drop-down, where the list always displays, a drop-down list box, where the text portion is not editable and you must select an arrow to view the drop-down list box, or the default drop-down list box, where the text portion is editable and the user must press the arrow key to view the list. To always display a list that the user cannot edit, use a **System.Windows.Forms.ListBox** control.

**b) ComboBox**

*Example Syntax:*

**c) ToString**

[C#]	public	ComboBox();
[C++]	public:	ComboBox();
[VB]	Public	Sub New()
[JScript]	public	function ComboBox();

*Description*

Initializes a new instance of the **System.Windows.Forms.ComboBox** class.

Creates a new ComboBox control. The default style for the combo is a regular DropDown Combo.

**d) AccessibilityObject**

**e) AccessibleDefaultActionDescription**

**f) AccessibleDescription**

**g) AccessibleName**

**h) AccessibleRole**

**i) AllowDrop**

**j) Anchor**

**k) BackColor**

**l) ToString**

*Description*

1 The background color of this control. This is an ambient property and will  
always return a non-null value.

2 *m) BackgroundImage*

3 *n) ToString*

4  
5 [C#] public override Image BackgroundImage {get; set;}

6 [C++] public: \_\_property virtual Image\* get\_BackgroundImage();public:  
7 \_\_property virtual void set\_BackgroundImage(Image\*);

8 [VB] Overrides Public Property BackgroundImage As Image

9 [JScript] public function get BackgroundImage() : Image;public function set  
10 BackgroundImage(Image);

11  
12 *Description*

- o) *BindingContext*
- p) *Bottom*
- q) *Bounds*
- r) *CanFocus*
- s) *CanSelect*
- t) *Capture*
- u) *CausesValidation*
- v) *ClientRectangle*
- w) *ClientSize*
- x) *CompanyName*
- y) *Container*
- z) *ContainsFocus*
- aa) *ContextMenu*
- bb) *Controls*
- cc) *Created*
- dd) *CreateParams*
- ee) *ToString*

## *Description*

Returns the parameters needed to create the handle. Inheriting classes can override this to provide extra functionality. They should not, however, forget to call `base.CreateParams()` first to get the struct filled up with the basic info.

- ff) Cursor*
- gg) DataBindings*
- hh) DataManager*
- ii) DataSource*
- jj) DefaultImeMode*
- kk) DefaultSize*
- ll) ToString*

#### *Description*

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

- mm) DesignMode*
- nn) DisplayMember*
- oo) DisplayRectangle*
- pp) Disposing*
- qq) Dock*
- rr) DrawMode*
- ss) ToString*

#### *Description*

Gets or sets a value indicating whether your code or the operating system will handle drawing of elements in the list.



1            *tt)    DropDownStyle*

2            *uu)    ToString*

3  
4 [C#]       public       ComboBoxStyle    DropDownStyle       {get;       set;}  
5 [C++]    public:    \_\_property    ComboBoxStyle    get\_DropDownStyle();public:  
6    \_\_property               void               set\_DropDownStyle(ComboBoxStyle);  
7 [VB]       Public       Property       DropDownStyle       As       ComboBoxStyle  
8 [JScript] public function get DropDownStyle() : ComboBoxStyle;public function  
9 set                               DropDownStyle(ComboBoxStyle);

10  
11 *Description*

12 Gets or sets a value specifying the style of the combo box.

13 The **System.Windows.Forms.ComboBox.DropDownStyle** property controls  
14 the interface that is presented to the user. You can enter a value that allows for  
15 a simple drop-down list box, where the list always displays, a drop-down list box,  
16 where the text portion is not editable and you must select an arrow to view the  
17 drop down, or the default drop-down list box, where the text portion is editable  
18 and the user must press the arrow key to view the list. To always display a list  
19 that the user cannot edit, use a **System.Windows.Forms.ListBox** control.

17            *vv)    DropDownWidth*

18            *ww)    ToString*

19  
20 [C#]       public       int       DropDownWidth       {get;       set;}  
21 [C++]    public:    \_\_property    int    get\_DropDownWidth();public:    \_\_property    void  
22    set\_DropDownWidth(int);  
23 [VB]       Public       Property       DropDownWidth       As       Integer  
24 [JScript] public function get DropDownWidth() : int;public function set  
25 DropDownWidth(int);

## Description

Gets or sets the width of the of the drop-down portion of a combo box.

If a value has not been set for the

**System.Windows.Forms.ComboBox.DropDownWidth** , this property returns the **System.Windows.Forms.Control.Width** of the combo box.

*xx) DroppedDown*

*yy) ToString*

```
[C#]      public      bool      DroppedDown      {get;      set;}
```

```
[C++] public: __property bool get_DroppedDown();public: __property void  
set_DroppedDown(bool);
```

```
[VB]      Public      Property      DroppedDown      As      Boolean
```

```
[JScript] public function get DroppedDown() : Boolean;public function set  
DroppedDown(Boolean);
```

## Description

Gets or sets a value indicating whether the combo box is displaying its drop-down portion.

*zz) Enabled*

*aaa) Events*

*bbb) Focused*

*ccc) ToString*

## Description

Gets a value indicating whether the **System.Windows.Forms.ComboBox** has focus.

*ddd) Font*

*eee) FontHeight*

*fff) ForeColor*

*ggg) ToString*

#### *Description*

Gets or sets the foreground color of the control.

*hhh) Handle*

*iii) HasChildren*

*jjj) Height*

*kkk) ImeMode*

*lll) IntegralHeight*

*mmm) ToString*

#### *Description*

Gets or sets a value indicating whether the control should resize to avoid showing partial items.

When this property is set to **true** , the control automatically resizes to ensure that an item is not partially displayed. If you want to maintain the original size of the **System.Windows.Forms.ComboBox** based on the space requirements of your form, set this property to **false** . If the **System.Windows.Forms.ComboBox** does not contain any items, this property has no effect.

1        *nnn) InvokeRequired*

2        *ooo) IsAccessible*

3        *ppp) IsDisposed*

4        *qqq) IsHandleCreated*

5        *rrr) ItemHeight*

6        *sss) ToString*

7  
8  
9        *Description*

10       Gets the height of an item in the combo box.

11       When the **System.Windows.Forms.ComboBox.DrawMode** property is set to  
12       **System.Windows.Forms.DrawMode.OwnerDrawFixed** , all items have the  
13       same height. When the **System.Windows.Forms.DrawMode** property is set  
14       to **System.Windows.Forms.DrawMode.OwnerDrawVariable** , the  
15       **System.Windows.Forms.ComboBox.ItemHeight** property specifies the  
16       height of each item added to the **System.Windows.Forms.ComboBox** .  
17       Because each item in an owner-drawn list can have a different height, you can  
18       use the

19       **System.Windows.Forms.ComboBox.GetItemHeight(System.Int32)**  
20       method to get the height of a specific item in the  
21       **System.Windows.Forms.ComboBox** . If you use the  
22       **System.Windows.Forms.ComboBox.ItemHeight** property on a  
23       **System.Windows.Forms.ComboBox** with items of variable height, this  
24       property returns the height of the first item in the control.

19       *ttt) Items*

20       *uuu) ToString*

22       [C#]        public        ComboBox.ObjectCollection        Items        {get;}

23       [C++]       public:       \_\_property       ComboBox.ObjectCollection\*       get\_Items();

24       [VB]       Public       ReadOnly       Property       Items       As       ComboBox.ObjectCollection

```
[JScript] public function get Items() : ComboBox.ObjectCollection;
```

### *Description*

Gets an object representing the collection of the items contained in this **System.Windows.Forms.ComboBox** .

This property enables you to obtain a reference to the list of items that are currently stored in the **System.Windows.Forms.ComboBox** . With this reference, you can add items, remove items, and obtain a count of the items in the collection. For more information on the tasks that can be performed with the item collection, see the **System.Windows.Forms.ComboBox.ObjectCollection** class reference topics.

*vvv) Left*

*www) Location*

*xxx) MaxDropDownItems*

*yyy) ToString*

### *Description*

Gets or sets the maximum number of items to be shown in the drop-down portion of the **System.Windows.Forms.ComboBox** .

*zzz) MaxLength*

*aaaa) ToString*

```
[C#] public int MaxLength {get; set;}
```

```
[C++] public: __property int get_MaxLength();public: __property void  
set_MaxLength(int);
```

```
[VB] Public Property MaxLength As Integer
```

[JScript] public function get MaxLength() : int; public function set  
MaxLength(int);

*Description*

Gets or sets the maximum number of characters allowed in the editable portion  
of a combo box.

*bbbb) Name*

*cccc) Parent*

*dddd) PreferredHeight*

*eeee) ToString*

*Description*

Gets the preferred height of the **System.Windows.Forms.ComboBox** .

The preferred height is a value based on the font height and an adjustment for  
the border.

*ffff) ProductName*

*gggg) ProductVersion*

*hhhh) RecreatingHandle*

*iiii) Region*

*jjjj) RenderRightToLeft*

*kkkk) ResizeRedraw*

*llll) Right*

*mmmm)RightToLeft*

*nnnn) SelectedIndex*

*oooo) ToString*

### *Description*

Gets or sets the index specifying the currently selected item.

This property indicates the zero-based index of the currently selected item in the combo box list. Setting a new index raises the **System.Windows.Forms.ComboBox.SelectedIndexChanged** event.

*pppp) SelectedItem*

*qqqq) ToString*

[C#]            public            object            SelectedItem            {get;            set;}

[C++] public: \_\_property Object\* get\_SelectedItem();public: \_\_property void  
set\_SelectedItem(Object\*);

[VB]            Public            Property            SelectedItem            As            Object

[JScript] public function get SelectedItem() : Object;public function set

SelectedItem(Object);

### Description

Gets or sets currently selected item in the **System.Windows.Forms.ComboBox** .

When you set the **System.Windows.Forms.ComboBox.SelectedItem** property to an object, the **System.Windows.Forms.ComboBox** attempts to make that object the currently selected one in the list. If the object is found in the list, it is displayed in the edit portion of the **System.Windows.Forms.ComboBox** and the **System.Windows.Forms.ComboBox.SelectedIndex** property is set to the corresponding index. If the object does not exist in the list the **System.Windows.Forms.ComboBox.SelectedIndex** property is left at its current value.

*rrrr) SelectedText*

*ssss) ToString*

[C#]            public            string            SelectedText            {get;            set;}

[C++] public: \_\_property String\* get\_SelectedText();public: \_\_property void set\_SelectedText(String\*);

[VB]            Public            Property            SelectedText            As            String

[JScript] public function get SelectedText() : String;public function set SelectedText(String);

### Description

Gets or sets the text that is selected in the editable portion of a **System.Windows.Forms.ComboBox** .

You can assign text to this property to change the text currently selected in the combo box. If no text is currently selected in the combo box, this property returns a zero-length string.



1        *tttt) SelectedValue*

2        *uuuu) SelectionLength*

3        *vvvv) ToString*

4  
5  
6        *Description*

7        Gets or sets the number of characters selected in the editable portion of the  
8        combo box.

9        You can use this property to determine whether any characters are currently  
10       selected in the combo box control before performing operations on the selected  
11       text. When the value of the

12       **System.Windows.Forms.ComboBox.SelectionLength** property is set to a  
13       value that is larger than the number of characters within the text of the control,  
14       the value of the **System.Windows.Forms.ComboBox.SelectionLength**  
15       property is set to the entire length of text within the control minus the value of  
16       the **System.Windows.Forms.ComboBox.SelectionStart** property (if any  
17       value is specified for the

18       **System.Windows.Forms.ComboBox.SelectionStart** property).

19        *www)SelectionStart*

20        *xxxx) ToString*

21        [C#]        public        int        SelectionStart        {get;        set;}

22        [C++]    public: \_\_property int get\_SelectionStart();public: \_\_property void  
23        set\_SelectionStart(int);

24        [VB]        Public        Property        SelectionStart        As        Integer

25        [JScript] public function get SelectionStart() : int;public function set  
SelectionStart(int);

26        *Description*

Gets or sets the starting index of text selected in the combo box.

If no text is selected in the control, this property indicates the insertion point for new text. If you set this property to a location beyond the length of the text in the control, the selection start position is placed after the last character. When text is selected in the text box control, changing this property can release the value of the **System.Windows.Forms.ComboBox.SelectionLength** property. If the remaining text in the control after the position indicated by the **System.Windows.Forms.ComboBox.SelectionStart** property is less than the value of the **System.Windows.Forms.ComboBox.SelectionLength** property, the value of the **System.Windows.Forms.ComboBox.SelectionLength** property is automatically decreased. The value of the **System.Windows.Forms.ComboBox.SelectionStart** property never causes an increase in the **System.Windows.Forms.ComboBox.SelectionLength** property.

*yyyy) ShowFocusCues*

*zzzz) ShowKeyboardCues*

*aaaaa) Site*

*bbbbbb) Size*

*ccccc) Sorted*

*dddd) ToString*

### *Description*

Gets or sets a value indicating whether the items in the combo box are sorted.

This property specifies whether the **System.Windows.Forms.ComboBox** sorts existing entries and add new entries to the appropriate sorted position in the list. You can use this property to automatically sort items in a **System.Windows.Forms.ComboBox**. As items are added to a sorted **System.Windows.Forms.ComboBox**, the items are moved to the appropriate location in the sorted list. When you set the property to **false**, new items are added to the end of the existing list. The sort is case-insensitive and in alphabetically ascending order.

1        *eeeeee) TabIndex*

2        *ffffff) TabStop*

3        *ggggg) Tag*

4        *hhhhh)Text*

5        *iiii) ToString*

6  
7  
8        *Description*

9        Gets or sets the text associated with this control.

10       When setting the **System.Windows.Forms.ComboBox.Text** property, **null** or  
11       an empty string("") sets the  
12       **System.Windows.Forms.ComboBox.SelectedIndex** to -1.

13        *jjjjj) Top*

14        *kkkkk) TopLevelControl*

15        *lllll) ValueMember*

16        *mmmmm)Visible*

17        *nnnnn)Width*

18        *ooooo) WindowTarget*

19        *ppppp) ToString*

20  
21  
22        *Description*

23        Occurs when a visual aspect of an owner-drawn  
24        **System.Windows.Forms.ComboBox** changes.

25        This event is used by an owner-drawn **System.Windows.Forms.ComboBox** .  
      You can use this event to perform the tasks needed to draw items in the  
      **System.Windows.Forms.ComboBox** . If you have a variable sized item

(when the **System.Windows.Forms.ComboBox.DrawMode** property set to **System.Windows.Forms.DrawMode.OwnerDrawVariable** ), before drawing an item, the **System.Windows.Forms.ComboBox.MeasureItem** event is raised. You can create an event handler for the **System.Windows.Forms.ComboBox.MeasureItem** event to specify the size for the item that you are going to draw in your event handler for the **System.Windows.Forms.ComboBox.DrawItem** event.

*qqqqq) ToString*

[C#]	public	event	EventHandler	DropDown;
[C++]	public:	__event	EventHandler*	DropDown;
[VB]	Public	Event	DropDown	As EventHandler

#### *Description*

Occurs when the drop-down portion of a **System.Windows.Forms.ComboBox** is shown.

For more information about handling events, see .

*rrrrr) ToString*

[C#]	public	event	EventHandler	DropDownStyleChanged;
[C++]	public:	__event	EventHandler*	DropDownStyleChanged;
[VB]	Public	Event	DropDownStyleChanged	As EventHandler

#### *Description*

Occurs when the **System.Windows.Forms.ComboBox.DropDownStyle** property has changed.

For more information about handling events, see .

sssss) *ToString*

*Description*

Occurs when an owner-drawn **System.Windows.Forms.ComboBox** is created and the sizes of the list items are determined.

You can create an event handler for this event to specify the size an item is made before it is drawn in the

**System.Windows.Forms.ComboBox.DrawItem** event.

tttt) *ToString*

*Description*

Occurs when the **System.Windows.Forms.ComboBox.SelectedIndex** property has changed.

You can create an event handler for this event to determine when the selected index in the **System.Windows.Forms.ComboBox** has been changed. This can be useful when you need to display information in other controls based on the current selection in the **System.Windows.Forms.ComboBox**. You can use the event handler for this event to load the information in the other controls.

uuuuu) *ToString*

*Description*

Occurs when the selected item has changed and that change is committed.

You can create a

**System.Windows.Forms.ComboBox.SelectionChangeCommitted** event handler to provide special handling for the

**System.Windows.Forms.ComboBox** when the user changes the selected item in the list.

### vvvvv) *AddItemsCore*

```
[C#]    protected    virtual    void    AddItemsCore(object[]    value);
[C++]    protected:    virtual    void    AddItemsCore(Object*    value    __gc[]);
[VB]    Overridable    Protected    Sub    AddItemsCore(ByVal    value()    As    Object)
[JScript]    protected    function    AddItemsCore(value    :    Object[]);
```

#### *Description*

Adds the specified items to the combo box.

The

**System.Windows.Forms.ComboBox.AddItemsCore(System.Object[])** method allows derived classes to provide special behaviour when adding items. An array of **System.Object** to append to the **System.Windows.Forms.ComboBox** .

### wwwww) *BeginUpdate*

```
[C#]                public                void                BeginUpdate();
[C++]                public:                void                BeginUpdate();
[VB]                Public                Sub                BeginUpdate()
[JScript]                public                function                BeginUpdate();
```

#### *Description*

Maintains performance when items are added to the **System.Windows.Forms.ComboBox** one at a time.

This method prevents the control from painting until the **System.Windows.Forms.ComboBox.EndUpdate** method is called.

## xxxxx) *Dispose*

[C#]	protected	override	void	Dispose(bool	disposing);
[C++]	protected:		void	Dispose(bool	disposing);
[VB]	Overrides	Protected	Sub	Dispose(ByVal	disposing As Boolean)
[JScript]	protected	override	function	Dispose(disposing	: Boolean);

### *Description*

## yyyyy) *EndUpdate*

[C#]	public	void	EndUpdate();
[C++]	public:	void	EndUpdate();
[VB]	Public	Sub	EndUpdate()
[JScript]	public	function	EndUpdate();

### *Description*

Resumes painting the **System.Windows.Forms.ComboBox** control after painting is suspended by the **System.Windows.Forms.ComboBox.BeginUpdate** method.

The preferred way to add items to the **System.Windows.Forms.ComboBox** is to use the **System.Windows.Forms.ComboBox.ObjectCollection.AddRange(System.Object[])** method of the **System.Windows.Forms.ComboBox.ObjectCollection** class (through the **System.Windows.Forms.ComboBox.Items** property of the **System.Windows.Forms.ComboBox** ). This enables you to add an array of items to the list at one time. However, if you want to add items one at a time using the **System.Windows.Forms.ComboBox.ObjectCollection.Add(System.Object)** method of the **System.Windows.Forms.ComboBox.ObjectCollection**

class, you can use the **System.Windows.Forms.ComboBox.BeginUpdate** method to prevent the control from repainting the **System.Windows.Forms.ComboBox** each time an item is added to the list. Once you have completed the task of adding items to the list, call the **System.Windows.Forms.ComboBox.EndUpdate** method to enable the **System.Windows.Forms.ComboBox** to repaint. This way of adding items can prevent flickered drawing of the **System.Windows.Forms.ComboBox** when a large number of items are being added to the list.

#### zzzzz) FindString

[C#]                public                int                FindString(string                s);

[C++]              public:              int              FindString(String\*              s);

[VB]   Public   Function   FindString(ByVal   s   As   String)   As   Integer

[JScript] public function FindString(s : String) : int; Finds the first item in the **System.Windows.Forms.ComboBox** that starts with the specified string.

#### Description

Finds the first item in the combo box that starts with the specified string.

**Return Value:** The zero-based index of the first item found; returns -1 if no match is found.

The search performed by this method is not case-sensitive. The *s* parameter is a substring to compare against the text associated with the items in the combo box list. The search performs a partial match starting from the beginning of the text, and returning the first item in the list that matches the specified substring. You can then perform tasks, such as removing the item that contains the search text using the

**System.Windows.Forms.ComboBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. Once you have found the specified text, if you want to search for other instances of the text in the **System.Windows.Forms.ComboBox**, you must use the version of the **System.Windows.Forms.ComboBox.FindString(System.String)** method that provides a parameter for specifying a starting index within the **System.Windows.Forms.ComboBox**. If you want to perform a search for an exact word match instead of a partial match, use the **System.Windows.Forms.ComboBox.FindStringExact(System.String)** method. The **System.String** to search for.



***aaaaaa)FindString***

```
[C#] public int FindString(string s, int startIndex);
```

```
[C++] public: int FindString(String* s, int startIndex);
```

[VB] Public Function FindString(ByVal s As String, ByVal startIndex As Integer)

As	Integer
----	---------

```
[JScript] public function FindString(s : String, startIndex : int) : int;
```

*Description*

Finds the first item after the given index which starts with the given string. The search is not case sensitive.

**Return Value:** The zero-based index of the first item found; returns -1 if no match is found.

The search performed by this method is not case-sensitive. The *s* parameter is a substring to compare against the text associated with the items in the combo box list. The search performs a partial match starting from the beginning of the text, returning the first item in the list that matches the specified substring. You can then perform tasks, such as removing the item that contains the search text using the

**System.Windows.Forms.ComboBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. This method is typically used after a call has been made using the version of this method that does not specify a starting index. Once an initial item has been found in the list, this method is typically used to find further instances of the search text by specifying the index position in the *startIndex* parameter of the item after the first found instance of the search text. If you want to perform a search for an exact word match instead of a partial match, use the

**System.Windows.Forms.ComboBox.FindStringExact(System.String)** method. The **System.String** to search for. The zero-based index of the item before the first item to be searched. Set to -1 to search from the beginning of the control.

***bbbbbb)FindStringExact***

```
[C#] public int FindStringExact(string s);
```

```

1 [C++]      public:      int      FindStringExact(String*      s);
2 [VB] Public Function FindStringExact(ByVal s As String) As Integer
3 [JScript] public function FindStringExact(s : String) : int; Finds the item that
4 exactly      matches      the      specified      string.

```

### 6 *Description*

7 Finds the first item in the combo box that matches the specified string.

8 *Return Value:* The zero-based index of the first item found; returns -1 if no match is found.

9 The search performed by this method is not case-sensitive. The *s* parameter is a string to compare against the text associated with the items in the combo box list. The search looks for a match starting from the beginning of the text, returning the first item in the list that matches the specified substring. You can then perform tasks, such as removing the item that contains the search text using the

12 **System.Windows.Forms.ComboBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. Once you have found the specified text, if you want to search for other instances of the text in the

14 **System.Windows.Forms.ComboBox**, you must use the version of the **System.Windows.Forms.ComboBox.FindStringExact(System.String)**

15 method that provides a parameter for specifying a starting index within the **System.Windows.Forms.ComboBox**. If you want to perform partial word search instead of an exact word match, use the

17 **System.Windows.Forms.ComboBox.FindString(System.String)** method. The **System.String** to search for.

18 *cccccc)FindStringExact*

```

20 [C#]      public      int      FindStringExact(string      s,      int      startIndex);
21 [C++]      public:      int      FindStringExact(String*      s,      int      startIndex);
22 [VB] Public Function FindStringExact(ByVal s As String, ByVal startIndex As
23 Integer)                                     As Integer
24 [JScript] public function FindStringExact(s : String, startIndex : int) : int;
25

```

## Description

Finds the first item after the specified index that matches the specified string.

**Return Value:** The zero-based index of the first item found; returns -1 if no match is found.

The search performed by this method is not case-sensitive. The *s* parameter is a string to compare against the text associated with the items in the combo box list. The search looks for a match starting from the beginning of the text, returning the first item in the list that matches the specified substring. You can then perform tasks, such as removing the item that contains the search text using the

**System.Windows.Forms.ComboBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. This method is typically used after a call has been made using the version of this method that does not specify a starting index. Once an initial item has been found in the list, this method is typically used to find further instances of the search text by specifying the index position in the *startIndex* parameter of the item after the first found instance of the search text. If you want to perform partial word search instead of an exact word match, use the

**System.Windows.Forms.ComboBox.FindString(System.String)** method. The **System.String** to search for. The zero-based index of the item before the first item to be searched. Set to -1 to search from the beginning of the control.

## dddddd)GetItemHeight

[C#]            public            int            GetItemHeight(int            index);

[C++]            public:            int            GetItemHeight(int            index);

[VB]    Public    Function    GetItemHeight(ByVal index As Integer) As Integer

[JScript]    public    function    GetItemHeight(index    :    int)    :    int;

## Description

Returns the height of an item in the **System.Windows.Forms.ComboBox**.

**Return Value:** The height, in pixels, of the item at the specified index.

If the **System.Windows.Forms.ComboBox.DrawMode** property is not set to **System.Windows.Forms.DrawMode.OwnerDrawVariable**, the value of

the index parameter is ignored because all items in a standard **System.Windows.Forms.ComboBox** are the same size. You can use this property when you are using an owner-drawn **System.Windows.Forms.ComboBox** to determine the size of any item within the **System.Windows.Forms.ComboBox**. The index of the item to return the height of.

#### *eeeeee)IsInputKey*

[C#]      protected      override      bool      IsInputKey(Keys      keyData);  
 [C++]      protected:      bool      IsInputKey(Keys      keyData);  
 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As Boolean  
 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

#### *Description*

Determines if keyData is in input key that the control wants. Overridden to return true for RETURN and ESCAPE when the combo box is dropped down.

#### *ffffff) OnBackColorChanged*

[C#]      protected      override      void      OnBackColorChanged(EventArgs      e);  
 [C++]      protected:      void      OnBackColorChanged(EventArgs\*      e);  
 [VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)  
 [JScript] protected override function OnBackColorChanged(e : EventArgs);

#### *Description*

Indicates that a critical property, such as color or font has changed.

### **gggggg)OnDataSourceChanged**

[C#] protected override void OnDataSourceChanged(EventArgs e);  
[C++] protected: void OnDataSourceChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnDataSourceChanged(ByVal e As EventArgs)  
[JScript] protected override function OnDataSourceChanged(e : EventArgs);

#### *Description*

### **hhhhhh)OnDisplayMemberChanged**

[C#] protected override void OnDisplayMemberChanged(EventArgs e);  
[C++] protected: void OnDisplayMemberChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnDisplayMemberChanged(ByVal e As EventArgs)  
[JScript] protected override function OnDisplayMemberChanged(e : EventArgs);

#### *Description*

### **iiiiii) OnDrawItem**

[C#] protected virtual void OnDrawItem(DrawItemEventArgs e);  
[C++] protected: virtual void OnDrawItem(DrawItemEventArgs\* e);  
[VB] Overridable Protected Sub OnDrawItem(ByVal e As DrawItemEventArgs)  
[JScript] protected function OnDrawItem(e : DrawItemEventArgs);

## Description

Raises the **System.Windows.Forms.ComboBox.DrawItem** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DrawItemEventArgs** that contains the event data.

### *jjjjj) OnDropDown*

[C#]      protected      virtual      void      OnDropDown(EventArgs e);

[C++]      protected:      virtual      void      OnDropDown(EventArgs\* e);

[VB]      Overridable      Protected      Sub      OnDropDown(ByVal e As EventArgs)

[JScript]      protected      function      OnDropDown(e : EventArgs);

## Description

Raises the **System.Windows.Forms.ComboBox.DropDown** event.

This event is raised each time the drop-down is displayed. An **System.EventArgs** that contains the event data.

### *kkkkk) OnDropDownStyleChanged*

[C#]      protected      virtual      void      OnDropDownStyleChanged(EventArgs e);

[C++]      protected:      virtual      void      OnDropDownStyleChanged(EventArgs\* e);

[VB]      Overridable      Protected      Sub      OnDropDownStyleChanged(ByVal e As EventArgs)

[JScript]      protected      function      OnDropDownStyleChanged(e : EventArgs);

## Description

1 Raises the **System.Windows.Forms.ComboBox.DropDownStyleChanged**  
event.

2 This event is raised when you set  
3 **System.Windows.Forms.ComboBox.DropDownStyle** to a new value. An  
**System.EventArgs** that contains the event data.

4 *lllll) OnFontChanged*

6 [C#] protected override void OnFontChanged(EventArgs e);

7 [C++] protected: void OnFontChanged(EventArgs\* e);

8 [VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

9 [JScript] protected override function OnFontChanged(e : EventArgs);

11 *Description*

12 Indicates that a critical property, such as color or font has changed.

13 *mmmmmm)OnForeColorChanged*

15 [C#] protected override void OnForeColorChanged(EventArgs e);

16 [C++] protected: void OnForeColorChanged(EventArgs\* e);

17 [VB] Overrides Protected Sub OnForeColorChanged(ByVal e As EventArgs)

18 [JScript] protected override function OnForeColorChanged(e : EventArgs);

20 *Description*

21 Indicates that a critical property, such as color or font has changed.

22 *nnnnnn)OnHandleCreated*

24 [C#] protected override void OnHandleCreated(EventArgs e);

1 [C++] protected: void OnHandleCreated(EventArgs\* e);

2 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

3 [JScript] protected override function OnHandleCreated(e : EventArgs);

4  
5 *Description*

6 Overridden to make sure all the items and styles get set up correctly. Inheriting  
7 classes should not forget to call base.OnHandleCreated() Overridden to make  
8 sure all the items and styles get set up correctly. Inheriting classes should not  
9 forget to call base.OnHandleCreated()

10 ***oooooo)OnHandleDestroyed***

11 [C#] protected override void OnHandleDestroyed(EventArgs e);

12 [C++] protected: void OnHandleDestroyed(EventArgs\* e);

13 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

14 [JScript] protected override function OnHandleDestroyed(e : EventArgs);

15  
16 *Description*

17 We need to un-subclasses everything here. Inheriting classes should not forget  
18 to call base.OnHandleDestroyed() We need to un-subclasses everything here.  
19 Inheriting classes should not forget to call base.OnHandleDestroyed()

20 ***pppppp)OnKeyPress***

21 [C#] protected override void OnKeyPress(KeyPressEventArgs e);

22 [C++] protected: void OnKeyPress(KeyPressEventArgs\* e);

23 [VB] Overrides Protected Sub OnKeyPress(ByVal e As KeyPressEventArgs)

24 [JScript] protected override function OnKeyPress(e : KeyPressEventArgs);



## Description

Key press event handler. Overridden to close up the combo box when the user presses RETURN or ESCAPE.

### *qqqqqq)OnMeasureItem*

[C#] protected virtual void OnMeasureItem(MeasureItemEventArgs e);

[C++] protected: virtual void OnMeasureItem(MeasureItemEventArgs\* e);

[VB] Overridable Protected Sub OnMeasureItem(ByVal e As MeasureItemEventArgs)

[JScript] protected function OnMeasureItem(e : MeasureItemEventArgs);

## Description

Raises the **System.Windows.Forms.ComboBox.MeasureItem** event.

This event is raised when you insert an item into the list. The **System.Windows.Forms.MeasureItemEventArgs** that was raised.

### *rrrrrr) OnParentBackColorChanged*

[C#] protected override void OnParentBackColorChanged(EventArgs e);

[C++] protected: void OnParentBackColorChanged(EventArgs\* e);

[VB] Overrides Protected Sub OnParentBackColorChanged(ByVal e As EventArgs)

[JScript] protected override function OnParentBackColorChanged(e : EventArgs);

## Description

This method is called by the parent control when any property changes on the parent. This can be overridden by inheriting classes, however they must call `base.OnParentPropertyChanged`. Property change information.

### *sssss) OnResize*

[C#]       protected       override       void       OnResize(EventArgs    e);

[C++]       protected:       void       OnResize(EventArgs\*       e);

[VB]   Overrides   Protected   Sub   OnResize(ByVal   e   As   EventArgs)

[JScript]   protected   override   function   OnResize(e   :   EventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.Resize** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *ttttt) OnSelectedIndexChanged*

[C#]   protected   override   void   OnSelectedIndexChanged(EventArgs   e);

[C++]   protected:   void   OnSelectedIndexChanged(EventArgs\*       e);

[VB]   Overrides   Protected   Sub   OnSelectedIndexChanged(ByVal   e   As   EventArgs)

[JScript]   protected   override   function   OnSelectedIndexChanged(e   :   EventArgs);

### *Description*

Raises the **System.Windows.Forms.ComboBox.SelectedIndexChanged** event.

This event is raised when you select a new item from the list. This event is raised when you set **System.Windows.Forms.ComboBox.SelectedIndex** . An **System.EventArgs** that contains the event data.

### *uuuuuu)OnSelectedItemChanged*

```
[C#]    protected    virtual    void    OnSelectedItemChanged(EventArgs e);
[C++]    protected:    virtual    void    OnSelectedItemChanged(EventArgs* e);
[VB]    Overridable Protected Sub OnSelectedItemChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnSelectedItemChanged(e : EventArgs);
```

#### *Description*

Raises the SelectedItemChanged event.

This event is raised when you select a new item from the list. This event is raised when you set **System.Windows.Forms.ComboBox.SelectedIndex** . An **System.EventArgs** that contains the event data.

### *vvvvvv)OnSelectionChangeCommitted*

```
[C#]    protected    virtual    void    OnSelectionChangeCommitted(EventArgs e);
[C++]    protected:    virtual    void    OnSelectionChangeCommitted(EventArgs* e);
[VB]    Overridable Protected Sub OnSelectionChangeCommitted(ByVal e As EventArgs)
[JScript]    protected    function    OnSelectionChangeCommitted(e : EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.ComboBox.SelectionChangeCommitted** event.

This event is raised when a new item is selected and that change to that item is completed. This event is also raised when you set **System.Windows.Forms.ComboBox.SelectedIndex** . An **System.EventArgs** that contains the event data.

## *wwwwww)RefreshItem*

```
[C#]      protected      override      void      RefreshItem(int      index);
[C++]      protected:      void      RefreshItem(int      index);
[VB]      Overrides      Protected      Sub      RefreshItem(ByVal      index      As      Integer)
[JavaScript]      protected      override      function      RefreshItem(index      :      int);
```

### *Description*

Reparses the object at the given index, getting new text string for it.

## *xxxxxx)Select*

```
[C#]      public      void      Select(int      start,      int      length);
[C++]      public:      void      Select(int      start,      int      length);
[VB]      Public      Sub      Select(ByVal      start      As      Integer,      ByVal      length      As      Integer)
[JavaScript]      public      function      Select(start      :      int,      length      :      int); Selects a range of text.
```

### *Description*

Selects a range of text in the editable portion of the **System.Windows.Forms.ComboBox** .

If you want to set the start position to the first character in the control's text, set the *start* parameter to zero. You can use this method to select a substring of text, such as when searching through the text of the control and replacing information. The position of the first character in the current text selection within the text box. The number of characters to select.

## *yyyyyy)SelectAll*

```
[C#]      public      void      SelectAll();
```

[C++]	public:	void	SelectAll();
[VB]	Public	Sub	SelectAll()
[JScript]	public	function	SelectAll();

#### Description

Selects all the text in the editable portion of the **System.Windows.Forms.ComboBox** .

#### zzzzzz) SetBoundsCore

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

#### Description

Sets the size and location of the **System.Windows.Forms.ComboBox** . The horizontal location in pixels of the control. The vertical location in pixels of the control. The width in pixels of the control. The height in pixels of the control. One of the **System.Windows.Forms.BoundsSpecified** values.

### ***aaaaaaa)SetItemCore***

```
[C#] protected override void SetItemCore(int index, object value);
[C++] protected: void SetItemCore(int index, Object* value);
[VB] Overrides Protected Sub SetItemCore(ByVal index As Integer, ByVal value
As Object)
[JScript] protected override function SetItemCore(index : int, value : Object);
```

#### ***Description***

### ***bbbbbbb)SetItemsCore***

```
[C#] protected override void SetItemsCore(object[] value);
[C++] protected: void SetItemsCore(Object* value __gc[]);
[VB] Overrides Protected Sub SetItemsCore(ByVal value() As Object)
[JScript] protected override function SetItemsCore(value : Object[]);
```

#### ***Description***

Performs the work of setting the specified items to the combobox Performs the work of setting the specified items to the combobox

### ***ccccccc)ToString***

```
[C#] public override string ToString();
[C++] public: String* ToString();
[VB] Overrides Public Function ToString() As String
```

1 [JScript] public override function ToString() : String;

3 *Description*

4 Provides some interesting info about this control in String form.

5 *Return Value:* String Provides some interesting info about this control in String form.

6 *ddddddd)WndProc*

8 [C#] protected override void WndProc(ref Message m);

9 [C++] protected: void WndProc(Message\* m);

10 [VB] Overrides Protected Sub WndProc(ByRef m As Message)

11 [JScript] protected override function WndProc(m : Message);

13 *Description*

14 The comboboxs window procedure. Inheritng classes can override this to add extra functionality, but should not forget to call base.wndProc(m); to ensure the combo continues to function properly. A Windows Message Object.

16 ComboBoxStyle enumeration (System.Windows.Forms)

17 a) *WndProc*

20 *Description*

21 Specifies the **System.Windows.Forms.ComboBox** style.

22 The **System.Windows.Forms.ComboBox.DropDownStyle** property determines whether the user can enter a new value in the text portion, and whether the list portion is always displayed.

**b) WndProc**

```
[C#]      public      const      ComboBoxStyle      DropDown;
[C++]      public:      const      ComboBoxStyle      DropDown;
[VB]      Public      Const      DropDown      As      ComboBoxStyle
[JScript]      public      var      DropDown      :      ComboBoxStyle;
```

*Description*

The text portion is editable. The user must click the arrow button to display the list portion.

**c) WndProc**

```
[C#]      public      const      ComboBoxStyle      DropDownList;
[C++]      public:      const      ComboBoxStyle      DropDownList;
[VB]      Public      Const      DropDownList      As      ComboBoxStyle
[JScript]      public      var      DropDownList      :      ComboBoxStyle;
```

*Description*

The user cannot directly edit the text portion. The user must click the arrow button to display the list portion.

**d) WndProc**

```
[C#]      public      const      ComboBoxStyle      Simple;
[C++]      public:      const      ComboBoxStyle      Simple;
[VB]      Public      Const      Simple      As      ComboBoxStyle
[JScript]      public      var      Simple      :      ComboBoxStyle;
```



## Description

The text portion is editable. The list portion is always visible.

CommonDialog class (System.Windows.Forms)

### a) ToString

## Description

Specifies the base class used for displaying dialog boxes on the screen.

Inherited classes are required to implement **System.Windows.Forms.CommonDialog.RunDialog(System.IntPtr)** by invoking **System.Windows.Forms.CommonDialog.ShowDialog** to create a specific common dialog box. Inherited classes can optionally override **System.Windows.Forms.CommonDialog.HookProc(System.IntPtr, System.IntPtr, System.IntPtr)** to implement specific dialog box hook functionality.

### b) CommonDialog

Example Syntax:

### c) ToString

[C#]	public	CommonDialog();
[C++]	public:	CommonDialog();
[VB]	Public	Sub New()
[JScript]	public	function CommonDialog();

## Description

Initializes a new instance of the **System.Windows.Forms.CommonDialog** class.

1 Initializes a new instance of the **System.Windows.Forms.CommonDialog**  
class.

2 *d) Container*

3 *e) DesignMode*

4 *f) Events*

5 *g) Site*

6 *h) ToString*

7  
8  
9  
10 *Description*

Occurs when the user clicks the Help button on a common dialog box.

For information about handling events, see .

11  
12  
13 *i) HookProc*

14  
15 [C#] protected virtual IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam,  
16 IntPtr lParam);

17 [C++] protected: virtual IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam,  
18 IntPtr lParam);

19 [VB] Overridable Protected Function HookProc(ByVal hWnd As IntPtr, ByVal  
20 msg As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As IntPtr

21 [JScript] protected function HookProc(hWnd : IntPtr, msg : int, wParam : IntPtr,  
22 lParam : IntPtr) : IntPtr;

23  
24 *Description*

25 Defines the common dialog box hook procedure that is overridden to add specific  
functionality to a common dialog box.

*Return Value:* A zero value if the default dialog box procedure processes the message; a nonzero value if the default dialog box procedure ignores the message.

A hook procedure allows the user to connect or insert other routines into a routine or application for the purpose of debugging or enhancing functionality. The handle to the dialog box window. The message being received. Additional information about the message. Additional information about the message.

*j) OnHelpRequest*

```
[C#]    protected    virtual    void    OnHelpRequest(EventArgs    e);
[C++]    protected:    virtual    void    OnHelpRequest(EventArgs*    e);
[VB]    Overridable Protected Sub OnHelpRequest(ByVal e As EventArgs)
[JScript]    protected    function    OnHelpRequest(e : EventArgs);
```

*Description*

Raises the **System.Windows.Forms.CommonDialog.HelpRequest** event.

This method is invoked when the Help button is clicked. Inheriting classes can override this method to handle the event. An

**System.Windows.Forms.HelpEventArgs** that provides the event data.

*k) OwnerWndProc*

```
[C#] protected virtual IntPtr OwnerWndProc(IntPtr hWnd, int msg, IntPtr wparam,
IntPtr lparam);
[C++] protected: virtual IntPtr OwnerWndProc(IntPtr hWnd, int msg, IntPtr
wparam, IntPtr lparam);
[VB] Overridable Protected Function OwnerWndProc(ByVal hWnd As IntPtr,
ByVal msg As Integer, ByVal wparam As IntPtr, ByVal lparam As IntPtr) As
IntPtr
```

```

1 [JScript] protected function OwnerWndProc(hWnd : IntPtr, msg : int, wParam :
2 IntPtr, lParam : IntPtr) : IntPtr;
3

```

#### 4 *Description*

5 Defines the owner window procedure that is overridden to add specific  
6 functionality to a common dialog box.

7 *Return Value:* The result of the message processing, which is dependent on the  
8 message sent.

9 Control is transferred here when messages are sent to the owner window of the  
10 common dialog. Inheriting classes can override this method to add specific  
11 functionality to a common dialog. The window handle of the message to send.  
12 The Win32 message to send. The *wparam* to send with the message. The *lparam*  
13 to send with the message.

#### 14 *l) Reset*

```

15 [C#]          public          abstract          void          Reset();
16 [C++]          public:          virtual          void          Reset()          =          0;
17 [VB]          MustOverride          Public          Sub          Reset()
18 [JScript]          public          abstract          function          Reset();
19

```

#### 20 *Description*

21 When overridden in a derived class, resets the properties of a common dialog to  
22 their default values.

23 Inheriting classes can override this method to reset their properties.

#### 24 *m) RunDialog*

```

25 [C#]          protected          abstract          bool          RunDialog(IntPtr          hwndOwner);
[C++]          protected:          virtual          bool          RunDialog(IntPtr          hwndOwner)          =          0;

```

[VB] MustOverride Protected Function RunDialog(ByVal hwndOwner As IntPtr)

As Boolean

[JScript] protected abstract function RunDialog(hwndOwner : IntPtr) : Boolean;

### *Description*

When overridden in a derived class, specifies a common dialog box.

*Return Value:* **true** if the dialog was successfully run; otherwise, **false** .

This method is invoked when the user of a common dialog calls

**System.Windows.Forms.CommonDialog.ShowDialog** , and it must be overridden by inherited classes of **System.Windows.Forms.CommonDialog** to implement a specific common dialog. An implementation of this method should store the *hwndOwner* and *hookProcPtr* parameters in the common dialog structure's **hwndOwner** and **lpfnHook** fields. A value that represents the window handle of the owner window for the common dialog box.

### *n) ShowDialog*

[C#] public DialogResult ShowDialog();

[C++] public: DialogResult ShowDialog();

[VB] Public Function ShowDialog() As DialogResult

[JScript] public function ShowDialog() : DialogResult; Runs a common dialog box.

### *Description*

Runs a common dialog box with a default owner.

*Return Value:* **System.Windows.Forms.DialogResult.OK** if the user clicks **OK** in the dialog box; otherwise, **System.Windows.Forms.DialogResult.Cancel**.

This method implements

**System.Windows.Forms.CommonDialog.RunDialog(System.IntPtr)** .

o) *ShowDialog*

```
[C#]    public    DialogResult    ShowDialog(IWin32Window    owner);
[C++]   public:    DialogResult    ShowDialog(IWin32Window*    owner);
[VB]    Public    Function    ShowDialog(ByVal    owner As IWin32Window) As
DialogResult
[JavaScript] public function ShowDialog(owner : IWin32Window) : DialogResult;
```

*Description*

Runs a common dialog box as a modal dialog with the specified owner.

**Return Value:** **System.Windows.Forms.DialogResult.OK** if the user clicks **OK** in the dialog box; otherwise, **System.Windows.Forms.DialogResult.Cancel**.

This version of the **System.Windows.Forms.CommonDialog.ShowDialog** method allows you to specify a specific form or control that will own the dialog box that is shown. If you use the version of this method that has no parameters, the dialog box being shown would be owned automatically by the currently active window of your application. Any object that implements **System.Windows.Forms.IWin32Window** that represents the top-level window that will own the modal dialog.

AxHost.ConnectionPointCookie class (System.Windows.Forms)

a) *ToString*

b) *AxHost.ConnectionPointCookie*

*Example Syntax:*

c) *ToString*

d) *Disconnect*

```
[C#]                public                void                Disconnect();
[C++]               public:                void                Disconnect();
```

[VB] Public Sub Disconnect()

[JScript] public function Disconnect(); Disconnect the current connection point. If the object is not connected, this method will do nothing.

e) *Finalize*

[C#] ~ConnectionPointCookie();

[C++] ~ConnectionPointCookie();

[VB] Overrides Protected Sub Finalize()

[JScript] protected override function Finalize();

ContainerControl class (System.Windows.Forms)

a) *ToString*

*Description*

Provides focus management functionality for controls that can function as a container for other controls.

A **System.Windows.Forms.ContainerControl** represents a control that can function as a container for other controls and provides focus management. Controls that inherit from this class can track the active control they contain, even when the focus moves somewhere within a different container.

b) *ContainerControl*

*Example Syntax:*

c) *ToString*

[C#] public ContainerControl();

[C++] public: ContainerControl();

[VB]	Public	Sub	New()
[JScript]	public	function	ContainerControl();

#### *Description*

Initializes a new instance of the **System.Windows.Forms.ContainerControl** class.

- d) AccessibilityObject*
- e) AccessibleDefaultActionDescription*
- f) AccessibleDescription*
- g) AccessibleName*
- h) AccessibleRole*
- i) ActiveControl*
- j) ToString*

#### *Description*

Gets or sets the active control on the container control.

This property activates or retrieves the active control on the container control.



- k) *AllowDrop*
- l) *Anchor*
- m) *AutoScroll*
- n) *AutoScrollMargin*
- o) *AutoScrollMinSize*
- p) *AutoScrollPosition*
- q) *BackColor*
- r) *BackgroundImage*
- s) *BindingContext*
- t) *ToString*

### *Description*

The binding manager for the container control.

- u) *Bottom*
- v) *Bounds*
- w) *CanFocus*
- x) *CanSelect*
- y) *Capture*
- z) *CausesValidation*
- aa) *ClientRectangle*
- bb) *ClientSize*
- cc) *CompanyName*
- dd) *Container*
- ee) *ContainsFocus*
- ff) *ContextMenu*
- gg) *Controls*
- hh) *Created*
- ii) *CreateParams*
- jj) *ToString*

*Description*

1	<b>kk)</b>	<b><i>Cursor</i></b>
2	<b>ll)</b>	<b><i>DataBindings</i></b>
3	<b>mm)</b>	<b><i>DefaultImeMode</i></b>
4	<b>nn)</b>	<b><i>DefaultSize</i></b>
5	<b>oo)</b>	<b><i>DesignMode</i></b>
6	<b>pp)</b>	<b><i>DisplayRectangle</i></b>
7	<b>qq)</b>	<b><i>Disposing</i></b>
8	<b>rr)</b>	<b><i>Dock</i></b>
9	<b>ss)</b>	<b><i>DockPadding</i></b>
10	<b>tt)</b>	<b><i>Enabled</i></b>
11	<b>uu)</b>	<b><i>Events</i></b>
12	<b>vv)</b>	<b><i>Focused</i></b>
13	<b>ww)</b>	<b><i>Font</i></b>
14	<b>xx)</b>	<b><i>FontHeight</i></b>
15	<b>yy)</b>	<b><i>ForeColor</i></b>
16	<b>zz)</b>	<b><i>Handle</i></b>
17	<b>aaa)</b>	<b><i>HasChildren</i></b>
18	<b>bbb)</b>	<b><i>Height</i></b>
19	<b>ccc)</b>	<b><i>HScroll</i></b>
20	<b>ddd)</b>	<b><i>ImeMode</i></b>
21	<b>eee)</b>	<b><i>InvokeRequired</i></b>
22	<b>fff)</b>	<b><i>IsAccessible</i></b>
23	<b>ggg)</b>	<b><i>IsDisposed</i></b>
24		
25		

*hhh) IsHandleCreated*

*iii) Left*

*jjj) Location*

*kkk) Name*

*lll) Parent*

*mmm) ParentForm*

*nnn) ToString*

*Description*

Gets or sets the form that the container control is assigned to.

**ooo) ProductName**  
**ppp) ProductVersion**  
**qqq) RecreatingHandle**  
**rrr) Region**  
**sss) RenderRightToLeft**  
**ttt) ResizeRedraw**  
**uuu) Right**  
**vvv) RightToLeft**  
**www) ShowFocusCues**  
**xxx) ShowKeyboardCues**  
**yyy) Site**  
**zzz) Size**  
**aaaa) TabIndex**  
**bbbb) TabStop**  
**cccc) Tag**  
**dddd) Text**  
**eeee) Top**  
**ffff) TopLevelControl**  
**gggg) Visible**  
**hhhh) VScroll**  
**iiii) Width**  
**jjjj) WindowTarget**  
**kkkk) AdjustFormScrollbars**

```

1
2 [C#] protected override void AdjustFormScrollbars(bool displayScrollbars);
3 [C++] protected: void AdjustFormScrollbars(bool displayScrollbars);
4 [VB] Overrides Protected Sub AdjustFormScrollbars(ByVal displayScrollbars As
5 Boolean)
6 [JScript] protected override function AdjustFormScrollbars(displayScrollbars :
7 Boolean);
8

```

### *Description*

#### **IIII) Dispose**

```

9
10
11
12 [C#] protected override void Dispose(bool disposing);
13 [C++] protected: void Dispose(bool disposing);
14 [VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)
15 [JScript] protected override function Dispose(disposing : Boolean);
16

```

### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.ContainerControl** .

#### Call

**System.Windows.Forms.ContainerControl.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.ContainerControl** . The **System.Windows.Forms.ContainerControl.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.ContainerControl** in an unusable state. After calling **System.Windows.Forms.ContainerControl.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.ContainerControl** so the memory it was occupying can be reclaimed by garbage collection.

### *mmmm)OnControlRemoved*

```
1  
2 [C#] protected override void OnControlRemoved(ControlEventArgs e);  
3 [C++] protected: void OnControlRemoved(ControlEventArgs* e);  
4 [VB] Overrides Protected Sub OnControlRemoved(ByVal e As  
5 ControlEventArgs)  
6 [JScript] protected override function OnControlRemoved(e : ControlEventArgs);  
7  
8
```

#### *Description*

Called when a child control is removed from this control. Contains the control that has just been removed.

### *nnnn) OnCreateControl*

```
12  
13 [C#] protected override void OnCreateControl();  
14 [C++] protected: void OnCreateControl();  
15 [VB] Overrides Protected Sub OnCreateControl()  
16 [JScript] protected override function OnCreateControl();  
17
```

#### *Description*

Raises the CreateControl event.

### *oooo) ProcessDialogChar*

```
21  
22 [C#] protected override bool ProcessDialogChar(char charCode);  
23 [C++] protected: bool ProcessDialogChar(__wchar_t charCode);  
24 [VB] Overrides Protected Function ProcessDialogChar(ByVal charCode As Char)  
25
```

As Boolean

[JScript] protected override function ProcessDialogChar(charCode : Char) : Boolean;

#### *Description*

Processes a dialog character. Overrides Control.processDialogChar(). This method calls the processMnemonic() method to check if the character is a mnemonic for one of the controls on the form. If processMnemonic() does not consume the character, then base.processDialogChar() is called.

*Return Value:* true to consume the character, false to allow further processing. character to pre-process.

#### *pppp) ProcessDialogKey*

[C#] protected override bool ProcessDialogKey(Keys keyData);

[C++] protected: bool ProcessDialogKey(Keys keyData);

[VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)

As Boolean

[JScript] protected override function ProcessDialogKey(keyData : Keys) :

Boolean;

#### *Description*

Processes a dialog key. Overrides Control.processDialogKey(). This method implements handling of the TAB, LEFT, RIGHT, UP, and DOWN keys in dialogs. The method performs no processing on keys that include the ALT or CONTROL modifiers. For the TAB key, the method selects the next control on the form.

*Return Value:* true to consume the key, false to allow further processing. key code and modifier flags.



### *qqqq) ProcessMnemonic*

```
[C#]    protected    override    bool    ProcessMnemonic(char    charCode);
[C++]    protected:    bool    ProcessMnemonic(__wchar_t    charCode);
[VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)
As                                             Boolean
[JScript] protected override function ProcessMnemonic(charCode : Char) :
Boolean;
```

### *rrrr) ProcessTabKey*

```
[C#]    protected    virtual    bool    ProcessTabKey(bool    forward);
[C++]    protected:    virtual    bool    ProcessTabKey(bool    forward);
[VB] Overridable Protected Function ProcessTabKey(ByVal forward As Boolean)
As                                             Boolean
[JScript] protected function ProcessTabKey(forward : Boolean) : Boolean;
```

### *Description*

Selects the next available control and makes it the active control.

*Return Value:* **true** if a control is selected; otherwise, **false** .

A control with its **System.Windows.Forms.Control.TabStop** property set to **false** cannot be selected, so the next available control will be selected. **true** to cycle forward through the controls in the **System.Windows.Forms.ContainerControl**; otherwise, **false**.

### *ssss) Select*

```
[C#]    protected    override    void    Select(bool    directed,    bool    forward);
[C++]    protected:    void    Select(bool    directed,    bool    forward);
```

[VB] Overrides Protected Sub Select(ByVal directed As Boolean, ByVal forward  
As Boolean)

[JScript] protected override function Select(directed : Boolean, forward :  
Boolean);

*tttt) IContainerControl.ActivateControl*

[C#] bool IContainerControl.ActivateControl(Control control);

[C++] bool IContainerControl::ActivateControl(Control\* control);

[VB] Function ActivateControl(ByVal control As Control) As Boolean  
Implements IContainerControl.ActivateControl

[JScript] function IContainerControl.ActivateControl(control : Control) : Boolean;

*uuuu) UpdateDefaultButton*

[C#] protected virtual void UpdateDefaultButton();

[C++] protected: virtual void UpdateDefaultButton();

[VB] Overridable Protected Sub UpdateDefaultButton()

[JScript] protected function UpdateDefaultButton();

### *Description*

Updates the default button based on current selection, and the acceptButton property.

*vvvv) Validate*

[C#] public bool Validate();

[C++] public: bool Validate();

[VB]	Public	Function	Validate()	As	Boolean
[JScript]	public	function	Validate()	:	Boolean;

#### Description

Validates the last invalidated control and its ancestors up through, but not including, the current control.

**Return Value:** **true** if validation is successful; otherwise, **false** .

#### www)WndProc

[C#]	protected	override	void	WndProc(ref	Message	m);
[C++]	protected:		void	WndProc(Message*		m);
[VB]	Overrides	Protected	Sub	WndProc(ByRef	m	As Message)
[JScript]	protected	override	function	WndProc(m	:	Message);

#### Description

ContentsResizedEventArgs class (System.Windows.Forms)

#### a) WndProc

#### Description

Provides data for the **System.Windows.Forms.RichTextBox.ContentsResized** event.

This event is raised when the bounding rectangle necessary to accept new text changes. If the text within the control spans multiple lines, the requested rectangle will always be the width of the control. You can handle this event in your control to implement auto-resizing for multiline

**System.Windows.Forms.RichTextBox** controls. The **System.Windows.Forms.ContentsResizedEventArgs** identifies the requested size of the **System.Windows.Forms.RichTextBox** .

**b) ContentsResizedEventArgs**

*Example Syntax:*

**c) WndProc**

```
[C#]      public      ContentsResizedEventArgs(Rectangle      newRectangle);  
[C++]     public:     ContentsResizedEventArgs(Rectangle      newRectangle);  
[VB]      Public      Sub      New(ByVal      newRectangle      As      Rectangle)  
[JScript] public function ContentsResizedEventArgs(newRectangle : Rectangle);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ContentsResizedEventArgs** class. A **System.Drawing.Rectangle** that specifies the requested dimensions of the **System.Windows.Forms.RichTextBox** control.

**d) NewRectangle**

**e) WndProc**

```
[C#]      public      Rectangle      NewRectangle      {get;}  
[C++]     public:     __property      Rectangle      get_NewRectangle();  
[VB]      Public      ReadOnly      Property      NewRectangle      As      Rectangle  
[JScript] public      function      get      NewRectangle()      :      Rectangle;
```

*Description*

Represents the requested size of the **System.Windows.Forms.RichTextBox** control.

When text is entered into the **System.Windows.Forms.RichTextBox** control, the **System.Windows.Forms.RichTextBox** control determines the proper size of the control in order to display all of the contents of the control. You can use

the **System.Windows.Forms.ContentResizedEventArgs.NewRectangle** property in an event handler for the **System.Windows.Forms.RichTextBox.ContentResized** event of the **System.Windows.Forms.RichTextBox** control to properly size the control to display all of the control's contents.

ContentResizedEventHandler delegate (System.Windows.Forms)

a) *ToString*

### *Description*

Represents the method that will handle the **System.Windows.Forms.RichTextBox.ContentResized** event of a **System.Windows.Forms.RichTextBox**.

When you create a **System.Windows.Forms.ContentResizedEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ContextMenu class (System.Windows.Forms)

a) *ToString*

### *Description*

Represents a shortcut menu.

The **System.Windows.Forms.ContextMenu** class represents shortcut menus that can be displayed when the user clicks the right mouse button over a control or area of the form. Shortcut menus are typically used to combine different menu items from a **System.Windows.Forms.MainMenu** of a form that are useful for the user given the context of the application. For example, you can use a shortcut menu assigned to a **System.Windows.Forms.TextBox** control to provide menu items for changing the font of the text, finding text within the control, or Clipboard features for copying and pasting text. You can also display new **System.Windows.Forms.MenuItem** objects in a shortcut menu that are not located within a **System.Windows.Forms.MainMenu** to provide situation

specific commands that are not appropriate for the **System.Windows.Forms.MainMenu** to display.

*b) ContextMenu*

*Example Syntax:*

*c) ToString*

[C#]	public	ContextMenu();
[C++]	public:	ContextMenu();
[VB]	Public	Sub New()

[JScript] public function ContextMenu(); Initializes a new instance of the **System.Windows.Forms.ContextMenu** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.ContextMenu** class with no menu items specified.

Once you have used this version of the constructor, you can add menu items to the **System.Windows.Forms.ContextMenu** by using the **System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)** method of the **System.Windows.Forms.Menu.MenuItemCollection** class. You can access the **System.Windows.Forms.Menu.MenuItemCollection** through the **System.Windows.Forms.Menu.MenuItems** property.

*d) ContextMenu*

*Example Syntax:*

*e) ToString*

[C#]	public	ContextMenu(MenuItem[]	menuItems);
[C++]	public:	ContextMenu(MenuItem*	menuItems[]);
[VB]	Public	Sub New(ByVal	menuItems() As MenuItem)

```
1 [JScript] public function ContextMenu(menuItems : MenuItem[]);
```

### 3 *Description*

4 Initializes a new instance of the **System.Windows.Forms.ContextMenu** class  
5 with a specified set of **System.Windows.Forms.MenuItem** objects.

6 You can use this version of the constructor to create a  
7 **System.Windows.Forms.ContextMenu** that has its menu items specified at  
8 the time it is created. Once you have used this version of the constructor, you  
9 can add additional menu items to the **System.Windows.Forms.ContextMenu**  
10 by using the  
11 **System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)**  
12 method of the **System.Windows.Forms.Menu.MenuItemCollection** class.  
13 You can access the **System.Windows.Forms.Menu.MenuItemCollection**  
14 through the **System.Windows.Forms.Menu.MenuItems** property. An array  
15 of **System.Windows.Forms.MenuItem** objects that represent the menu items  
16 to add to the context menu.

- 17 *f) Container*
- 18 *g) DesignMode*
- 19 *h) Events*
- 20 *i) Handle*
- 21 *j) IsParent*
- 22 *k) MdiListItem*
- 23 *l) MenuItems*
- 24 *m) RightToLeft*
- 25 *n) ToString*

### 24 *Description*

Gets or sets a value indicating whether text displayed by the control is displayed from right to left.

This property allows your menus to support languages that are written from right to left. When this property is set to **RightToLeft.Yes**, the menu item text is displayed from right to left instead of the default left to right method.

*o) Site*

*p) SourceControl*

*q) ToString*

#### *Description*

Gets the control that is displaying the shortcut menu.

This property enables you to determine which control currently displays the shortcut menu defined in the **System.Windows.Forms.ContextMenu**. If the context menu is not currently displayed, you can use this property to determine which control last displayed the shortcut menu. You can use this property in the **System.Windows.Forms.ContextMenu.Popup** event to ensure that the control displays the proper menu items. You can also use this property to pass a reference to the control to a method that performs the tasks associated with a menu command displayed in the shortcut menu. Since the **System.Windows.Forms.Form** class inherits from **System.Windows.Forms.Control**, you can also use this property if the **System.Windows.Forms.ContextMenu** is associated with a form.

*r) ToString*

#### *Description*

Occurs before the context menu is displayed.

You can use this event to initialize the **System.Windows.Forms.MenuItem** objects before they are displayed. For example, if you use a **System.Windows.Forms.ContextMenu** for three **System.Windows.Forms.TextBox** controls and you want to disable certain menu items in the **System.Windows.Forms.ContextMenu** depending on



which **System.Windows.Forms.TextBox** is displaying the shortcut menu, you can create an event handler for this event. You could use the **System.Windows.Forms.ContextMenu.SourceControl** property to determine which **System.Windows.Forms.TextBox** is about to display the **System.Windows.Forms.ContextMenu** and disable the appropriate **System.Windows.Forms.MenuItem** objects.

#### s) *OnPopup*

```
[C#]    protected    internal    virtual    void    OnPopup(EventArgs    e);
[C++]    protected    public:    virtual    void    OnPopup(EventArgs*    e);
[VB]    Overridable Protected Friend Dim Sub OnPopup(ByVal e As EventArgs)
[JScript]    package    function    OnPopup(e    :    EventArgs);
```

#### *Description*

Fires the popup event Fires the popup event Event object

#### t) *Show*

```
[C#]    public    void    Show(Control    control,    Point    pos);
[C++]    public:    void    Show(Control*    control,    Point    pos);
[VB]    Public Sub Show(ByVal control As Control, ByVal pos As Point)
[JScript]    public    function    Show(control    :    Control,    pos    :    Point);
```

#### *Description*

Displays the shortcut menu at the specified position.

Typically, a **System.Windows.Forms.ContextMenu** is displayed when the user clicks the right mouse button on a control or area of the form that the **System.Windows.Forms.ContextMenu** is bound to. You can use this method to manually display the shortcut menu at a specific location and bind it with a specific control. This method does not return until the menu is dismissed. A **System.Windows.Forms.Control** object that specifies the control with which

this context menu is associated. A **System.Drawing.Point** object that specifies the coordinates at which to display the menu. These coordinates are specified relative to the client coordinates of the control specified in the *control* parameter.

Control class (System.Windows.Forms)

*a) ToString*

*Description*

Defines the base class for controls, which are components with visual representation.

The **System.Windows.Forms.Control** class implements very basic functionality required by classes that display to the user. It handles user input through the keyboard and pointing devices. It handles message routing and security. It defines the bounds of a control (its position and size), although it does not implement painting. It provides a window handle (hWnd).

*b) Control*

*Example Syntax:*

*c) ToString*

[C#]	public	Control();
[C++]	public:	Control();
[VB]	Public	Sub New()

[JScript] public function Control(); Initializes a new instance of the **System.Windows.Forms.Control** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.Control** class with default settings.

The **System.Windows.Forms.Control** class is the base class for all controls used in a Windows Forms application. Because this class is not typically used to create an instance of the class, this constructor is typically not called directly but is instead called by a derived class.

*d) Control*

*Example Syntax:*

*e) ToString*

```
[C#]          public          Control(string          text);
[C++]          public:          Control(String*          text);
[VB]      Public      Sub      New(ByVal      text      As      String)
[JScript]      public      function      Control(text      :      String);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.Control** class with specific text. A **System.String** that represents the text displayed by the control.

*f) Control*

*Example Syntax:*

*g) ToString*

```
[C#]      public      Control(Control      parent,      string      text);
[C++]      public:      Control(Control*      parent,      String*      text);
[VB]      Public      Sub      New(ByVal      parent      As      Control,      ByVal      text      As      String)
[JScript]      public      function      Control(parent      :      Control,      text      :      String);
```

*Description*

1 Initializes a new instance of the **System.Windows.Forms.Control** class as a  
2 child control, with specific text. A **System.Windows.Forms.Control**  
3 representing the parent of this control. A **System.String** that represents the  
4 text displayed by the control.

5 *h) Control*

6 *Example Syntax:*

7 *i) ToString*

8 [C#] public Control(string text, int left, int top, int width, int height);

9 [C++] public: Control(String\* text, int left, int top, int width, int height);

10 [VB] Public Sub New(ByVal text As String, ByVal left As Integer, ByVal top As  
11 Integer, ByVal width As Integer, ByVal height As Integer)

12 [JScript] public function Control(text : String, left : int, top : int, width : int, height  
13 : int);

14 *Description*

15 Initializes a new instance of the **System.Windows.Forms.Control** class with  
16 specific text, size, and location. A **System.String** that represents the text  
17 displayed by the control. The **System.Drawing.Point.X** position of the control,  
18 in pixels, from the left edge of the control's container. The value is assigned to  
19 the **System.Windows.Forms.Control.Left** property. The  
20 **System.Drawing.Point.Y** position of the control, in pixels, from the top edge  
21 of the control's container. The value is assigned to the  
22 **System.Windows.Forms.Control.Top** property. The  
23 **System.Windows.Forms.Control.Width** of the control, in pixels. The  
24 **System.Windows.Forms.Control.Height** of the control, in pixels.

25 *j) Control*

*Example Syntax:*

k) *ToString*

[C#] public Control(Control parent, string text, int left, int top, int width, int height);

[C++] public: Control(Control\* parent, String\* text, int left, int top, int width, int height);

[VB] Public Sub New(ByVal parent As Control, ByVal text As String, ByVal left As Integer, ByVal top As Integer, ByVal width As Integer, ByVal height As Integer)

[JScript] public function Control(parent : Control, text : String, left : int, top : int, width : int, height : int);

*Description*

Initializes a new instance of the **System.Windows.Forms.Control** class as a child control, with specific text, size, and location. A **System.Windows.Forms.Control** representing the parent of the control. A **System.String** that represents the text displayed by the control. The **System.Drawing.Point.X** position of the control, in pixels, from the left edge of the control's container. The value is assigned to the **System.Windows.Forms.Control.Left** property. The **System.Drawing.Point.Y** position of the control, in pixels, from the top edge of the control's container. The value is assigned to the **System.Windows.Forms.Control.Top** property. The **System.Windows.Forms.Control.Width** of the control, in pixels. The **System.Windows.Forms.Control.Height** of the control, in pixels.

l) *AccessibilityObject*

m) *ToString*

[C#] public AccessibleObject AccessibilityObject {get;}

[C++] public: \_\_property AccessibleObject\* get\_AccessibilityObject();

1 [VB] Public ReadOnly Property AccessibilityObject As AccessibleObject

2 [JScript] public function get AccessibilityObject() : AccessibleObject;

3  
4 *Description*

5 Gets the **System.Windows.Forms.AccessibleObject** assigned to the control.

6 To control the instance returned from this method, override the  
**System.Windows.Forms.Control.CreateAccessibilityInstance** method.

7       *n)     AccessibleDefaultActionDescription*

8       *o)     ToString*

9  
10 [C#]     public     string     AccessibleDefaultActionDescription     {get;     set;}  
11

12 [C++]             public:             \_\_property             String\*  
13

14 get\_AccessibleDefaultActionDescription();public:             \_\_property             void  
15

16 set\_AccessibleDefaultActionDescription(String\*);  
17

18 [VB]     Public     Property     AccessibleDefaultActionDescription     As     String  
19

20 [JScript] public function get AccessibleDefaultActionDescription() : String;public  
21

22 function             set             AccessibleDefaultActionDescription(String);  
23

24  
25 *Description*

Gets or sets the default action description of the control for use by accessibility client applications.

An object's

**System.Windows.Forms.Control.AccessibleDefaultActionDescription** property describes the objects primary method of manipulation from the user's viewpoint. This property should be a verb or a short verb phrase.

p) *AccessibleDescription*

q) *ToString*

```
[C#]      public      string      AccessibleDescription      {get;      set;}
[C++] public: __property String* get_AccessibleDescription();public: __property
void                                             set_AccessibleDescription(String*);
[VB]      Public      Property      AccessibleDescription      As      String
[JScript] public function get AccessibleDescription() : String;public function set
AccessibleDescription(String);
```

#### *Description*

Gets or sets the description of the control used by accessibility client applications.

An object's **System.Windows.Forms.Control.AccessibleDescription** property provides a textual description about an object's visual appearance. The description is primarily used to provide greater context for low-vision or blind users, but can also be used for context searching or other applications.

r) *AccessibleName*

s) *ToString*

```
[C#]      public      string      AccessibleName      {get;      set;}
[C++] public: __property String* get_AccessibleName();public: __property void
set_AccessibleName(String*);
[VB]      Public      Property      AccessibleName      As      String
[JScript] public function get AccessibleName() : String;public function set
AccessibleName(String);
```

## Description

Gets or sets the name of the control used by accessibility client applications.

The **System.Windows.Forms.Control.AccessibleName** property is a label that briefly describes and identifies the object within its container, such as the text in a **System.Windows.Forms.Button**, the name of a **System.Windows.Forms.MenuItem**, or a label displayed next to a **System.Windows.Forms.TextBox** control.

t) *AccessibleRole*

u) *ToString*

```
[C#]      public      AccessibleRole      AccessibleRole      {get;      set;}
```

```
[C++] public: __property AccessibleRole get _AccessibleRole();public: __property  
void      set _AccessibleRole(AccessibleRole);
```

```
[VB]      Public      Property      AccessibleRole      As      AccessibleRole
```

```
[JScript] public function get AccessibleRole() : AccessibleRole;public function set  
AccessibleRole(AccessibleRole);
```

## Description

Gets or sets the accessible role of the control The **System.Windows.Forms.AccessibleRole** of the control. The default is **System.Windows.Forms.AccessibleRole.Default**.

The **System.Windows.Forms.Control.AccessibleRole** property describes what kind of user interface element an object is. If the control's role cannot be determined, the **System.Windows.Forms.Control.AccessibleRole** property is set to **System.Windows.Forms.AccessibleRole.Default**.



v) *AllowDrop*

w) *ToString*

```
[C#]      public      virtual      bool      AllowDrop      {get;      set;}
[C++] public: __property virtual bool get_AllowDrop();public: __property virtual
void                                             set_AllowDrop(bool);
[VB]      Overridable      Public      Property      AllowDrop      As      Boolean
[JScript] public function get AllowDrop() : Boolean;public function set
AllowDrop(Boolean);
```

### *Description*

Gets or sets a value indicating whether the control can accept data that the user drags onto it.

When overriding the **System.Windows.Forms.Control.AllowDrop** property in a derived class, use the base class's

**System.Windows.Forms.Control.AllowDrop** property to extend the base implementation. Otherwise, you must provide all the implementation. You are not required to override both the **get** and **set** methods of the **System.Windows.Forms.Control.AllowDrop** property; you can override only one if needed.

x) *Anchor*

y) *ToString*

```
[C#]      public      virtual      AnchorStyles      Anchor      {get;      set;}
[C++] public: __property virtual AnchorStyles get_Anchor();public: __property
virtual                                     void                                     set_Anchor(AnchorStyles);
[VB]      Overridable      Public      Property      Anchor      As      AnchorStyles
[JScript] public function get Anchor() : AnchorStyles;public function set
```

1 Anchor(AnchorStyles);

3 *Description*

4 Gets or sets which edges of the control are anchored to the edges of its container.

5 A control can be docked to one edge of its parent container or can dock to all  
6 edges and fill the parent container. For example, if you set this property to  
7 **System.Windows.Forms.DockStyle.Left**, the left edge of the control will be  
8 docked to the left edge of its parent control. Additionally, the docked edge of the  
control is resized to match that of its container control. Controls are docked in  
order of their z-order.

9 z) *BackColor*

10 aa) *ToString*

12 [C#] public virtual Color BackColor {get; set;}

13 [C++] public: \_\_property virtual Color get\_BackColor();public: \_\_property virtual  
14 void set\_BackColor(Color);

15 [VB] Overridable Public Property BackColor As Color

16 [JScript] public function get BackColor() : Color;public function set  
17 BackColor(Color);

19 *Description*

20 Gets or sets the background color for the control.

21 The **System.Windows.Forms.Control.BackColor** property does not support  
22 transparent colors unless the  
23 **System.Windows.Forms.ControlStyles.SupportsTransparentBackColor**  
style bit is set to **true**.

**bb) BackgroundImage**

**cc) ToString**

```
[C#]    public    virtual    Image    BackgroundImage    {get;    set;}
[C++]  public:  __property  virtual  Image*  get_BackgroundImage();public:
__property      virtual      void      set_BackgroundImage(Image*);
[VB]    Overridable  Public  Property  BackgroundImage  As  Image
[JScript] public function get BackgroundImage() : Image;public function set
BackgroundImage(Image);
```

*Description*

Gets or sets the background image displayed in the control.

Images with translucent or transparent colors are not supported by Windows Forms controls as background images.

**dd) BindingContext**

**ee) ToString**

```
[C#]    public    virtual    BindingContext    BindingContext    {get;    set;}
[C++]  public:  __property  virtual  BindingContext*  get_BindingContext();public:
__property      virtual      void      set_BindingContext(BindingContext*);
[VB]    Overridable  Public  Property  BindingContext  As  BindingContext
[JScript] public function get BindingContext() : BindingContext;public function
set                                     BindingContext(BindingContext);
```

*Description*

Gets or sets the **System.Windows.Forms.BindingContext** for the control.

The **System.Windows.Forms.BindingContext** object of a **System.Windows.Forms.Control** is used to return a single **System.Windows.Forms.BindingManagerBase** object for all data-bound controls contained by the **System.Windows.Forms.Control**. The **System.Windows.Forms.BindingManagerBase** object keeps all controls that are bound to the same data source synchronized. For example, setting the **System.Windows.Forms.BindingManagerBase.Position** property of the **System.Windows.Forms.BindingManagerBase** specifies the item in the underlying list that all data-bound controls point to.

*ff) Bottom*

*gg) ToString*

[C#]	public	int	Bottom	{get;}
[C++]	public:	__property	int	get_Bottom();
[VB]	Public	ReadOnly	Property	Bottom As Integer
[JScript]	public	function	get	Bottom() : int;

#### *Description*

Gets the distance between the bottom edge of the control and the top edge of its container's client area.

The value of this property is equal to the sum of the the **System.Windows.Forms.Control.Top** property value, plus the **System.Windows.Forms.Control.Height** property value.

*hh) Bounds*

*ii) ToString*

[C#]	public	Rectangle	Bounds	{get; set;}
[C++]	public:	__property	Rectangle	get_Bounds();public: __property void set_Bounds(Rectangle);
[VB]	Public	Property	Bounds	As Rectangle

```

1 [JScript] public function get Bounds() : Rectangle;public function set
2 Bounds(Rectangle);
3

```

#### *Description*

Gets or sets the size and location of the control.

*jj) CanFocus*

*kk) ToString*

```

9 [C#]          public          bool          CanFocus          {get;}
10 [C++]         public:         __property    bool          get_CanFocus();
11 [VB]         Public    ReadOnly    Property    CanFocus    As    Boolean
12 [JScript]     public    function    get    CanFocus()    :    Boolean;
13

```

#### *Description*

Gets a value indicating whether the control can receive focus.

In order for a control to receive input focus, the control must have a handle assigned to it, and the **System.Windows.Forms.Control.Visible** and **System.Windows.Forms.Control.Enabled** properties must both be set to **true**.

*ll) CanSelect*

*mm) ToString*

```

22 [C#]          public          bool          CanSelect          {get;}
23 [C++]         public:         __property    bool          get_CanSelect();
24 [VB]         Public    ReadOnly    Property    CanSelect    As    Boolean
25 [JScript]     public    function    get    CanSelect()    :    Boolean;

```

## Description

Gets a value indicating whether the control can be selected.

This property returns **true** if the control has **System.Windows.Forms.ControlStyles.Selectable** set to **true**, is contained in another control, and all of its parent controls are both visible and enabled.

*nn) Capture*

*oo) ToString*

[C#]            public            bool            Capture            {get;            set;}

[C++]   public:   \_\_property   bool   get\_Capture();public:   \_\_property   void  
set\_Capture(bool);

[VB]            Public            Property            Capture            As            Boolean

[JScript]   public   function   get   Capture()   :   Boolean;public   function   set  
Capture(Boolean);

## Description

Gets or sets a value indicating whether the control has captured the mouse.

When a control has captured the mouse, it receives mouse input whether or not the cursor is within its borders. The mouse is typically only captured during drag operations.

*pp) CausesValidation*

*qq) ToString*

[C#]            public            bool            CausesValidation            {get;            set;}

[C++]   public:   \_\_property   bool   get\_CausesValidation();public:   \_\_property   void

1 set\_CausesValidation(bool);

2 [VB] Public Property CausesValidation As Boolean

3 [JScript] public function get CausesValidation() : Boolean;public function set

4 CausesValidation(Boolean);

6 *Description*

7 Gets or sets a value indicating whether the control causes validation to be  
performed on all controls that require validation when it receives focus.

8 The **System.Windows.Forms.Control.CausesValidation** property value is  
9 typically set to **false** for controls such as a Help button.

10 *rr) ClientRectangle*

11 *ss) ToString*

13 [C#] public Rectangle ClientRectangle {get;}

14 [C++] public: \_\_property Rectangle get\_ClientRectangle();

15 [VB] Public ReadOnly Property ClientRectangle As Rectangle

16 [JScript] public function get ClientRectangle() : Rectangle;

18 *Description*

19 Gets the rectangle that represents the client area of the control.

20 The client area of a control is the bounds of the control, minus the non-client  
elements such as scroll bars, borders, title bars and menus.

22 *tt) ClientSize*

23 *uu) ToString*

25 [C#] public Size ClientSize {get; set;}

```

1 [C++] public: __property Size get_ClientSize();public: __property void
2 set_ClientSize(Size);

```

```

3 [VB] Public Property ClientSize As Size

```

```

4 [JScript] public function get ClientSize() : Size;public function set
5 ClientSize(Size);

```

### *Description*

Gets or sets the height and width of the client area of the control.

The client area of a control is the bounds of the control, minus the non-client elements such as scroll bars, borders, title bars and menus.

*vv) CompanyName*

*ww) ToString*

```

13 [C#] public string CompanyName {get;}

```

```

14 [C++] public: __property String* get_CompanyName();

```

```

15 [VB] Public ReadOnly Property CompanyName As String

```

```

16 [JScript] public function get CompanyName() : String;

```

### *Description*

Gets the name of the company or creator of the application containing the control.

The **System.Windows.Forms.Control.CompanyName** property is a read-only property. In order to change the value of this property, set the **System.Reflection.AssemblyCompanyAttribute.Company** property value of the **System.Reflection.AssemblyCompanyAttribute** .



xx) *Container*

yy) *ContainsFocus*

zz) *ToString*

#### *Description*

Gets a value indicating whether the control, or one of its child controls, currently has the input focus.

You can use this property to determine whether a control or any of the controls contained within it has the input focus. To determine whether the control has focus, regardless of whether any of its child controls have focus, use the **System.Windows.Forms.Control.Focused** property. To give a control the input focus, use the **System.Windows.Forms.Control.Focus** or **System.Windows.Forms.Control.Select** methods.

aaa) *ContextMenu*

bbb) *ToString*

```
[C#]    public    virtual    ContextMenu    ContextMenu    {get;    set;}
[C++]   public:   __property virtual ContextMenu* get_ContextMenu();public:
__property          virtual          void          set_ContextMenu(ContextMenu*);
[VB]    Overridable    Public    Property    ContextMenu    As    ContextMenu
[JavaScript] public function get ContextMenu() : ContextMenu;public function set
ContextMenu(ContextMenu);
```

#### *Description*

Gets or sets the shortcut menu associated with the control.

A shortcut menu is also known as a context menu. Shortcut menus are used to give context specific menu options to users when they right-click on the control.

ccc) *Controls*

ddd) *ToString*

```
[C#]      public      Control.ControlCollection      Controls      {get;}
[C++]     public:     __property      Control.ControlCollection*      get_Controls();
[VB]      Public      ReadOnly      Property      Controls      As      Control.ControlCollection
[JScript] public      function      get      Controls()      :      Control.ControlCollection;
```

#### *Description*

Gets the collection of controls contained within the control.

A **System.Windows.Forms.Control** can act as a parent to a collection of controls. For example, when several controls are added to a **System.Windows.Forms.Form**, each of the controls is a member of the **System.Windows.Forms.Control.ControlCollection** object assigned to the **System.Windows.Forms.Control.Controls** property of the form, which is derived from the **System.Windows.Forms.Control** class.

eee) *Created*

fff) *ToString*

```
[C#]      public      bool      Created      {get;}
[C++]     public:     __property      bool      get_Created();
[VB]      Public      ReadOnly      Property      Created      As      Boolean
[JScript] public      function      get      Created()      :      Boolean;
```

#### *Description*

Gets a value indicating whether the control has been created.

The **System.Windows.Forms.Control.Created** property returns **true** if the **System.Windows.Forms.Control** was successfully created even though the control's handle may not have been created or recreated yet.

*ggg) CreateParams*

*hhh) ToString*

```
[C#]    protected    virtual    CreateParams    CreateParams    {get;}
[C++]    protected:    __property    virtual    CreateParams*    get_CreateParams();
[VB]    Overridable Protected ReadOnly Property CreateParams As CreateParams
[JScript]    protected    function    get    CreateParams()    :    CreateParams;
```

### *Description*

Gets the required creation parameters when the control handle is created.

The **System.Windows.Forms.Control.CreateParams** property should not be overridden and used to adjust the properties of your derived control. Properties such as the **System.Windows.Forms.CreateParams.Caption** , **System.Windows.Forms.CreateParams.Width** , and **System.Windows.Forms.CreateParams.Height** should be set by the corresponding properties in your control such as **System.Windows.Forms.Control.Text** , **System.Windows.Forms.Control.Width** and **System.Windows.Forms.Control.Height** . The **System.Windows.Forms.CreateParams** object should only be extended when you are wrapping a standard Windows control class or to set styles not provided by the Windows Forms namespace. For more information about creating control parameters, see the **CreateWindow** and **CreateWindowEx** functions and the **CREATESTRUCT** structure documentation in the Windows Platform SDK reference located in the MSDN Library.

*iii) Cursor*

*jjj) ToString*

```
[C#]    public    virtual    Cursor    Cursor    {get;    set;}
```

```

1 [C++] public: __property virtual Cursor* get_Cursor();public: __property virtual
2 void set_Cursor(Cursor*);

```

```

3 [VB] Overridable Public Property Cursor As Cursor

```

```

4 [JScript] public function get Cursor() : Cursor;public function set Cursor(Cursor);

```

### 6 *Description*

7 Gets or sets the cursor that is displayed when the mouse pointer is over the control.

8 When overriding the **System.Windows.Forms.Control.Cursor** property in a  
9 derived class, use the base class's **System.Windows.Forms.Control.Cursor**  
10 property to extend the base implementation. Otherwise, you must provide all the  
11 implementation. You are not required to override both the **get** and **set** methods  
12 of the **System.Windows.Forms.Control.Cursor** property; you can override  
13 only one if needed.

### 12 *kkk) DataBindings*

### 13 *lll) ToString*

```

15 [C#] public ControlBindingsCollection DataBindings {get;}

```

```

16 [C++] public: __property ControlBindingsCollection* get_DataBindings();

```

```

17 [VB] Public ReadOnly Property DataBindings As ControlBindingsCollection

```

```

18 [JScript] public function get DataBindings() : ControlBindingsCollection;

```

### 20 *Description*

21 Gets the data bindings for the control.

22 Use the **System.Windows.Forms.Control.DataBindings** property to access  
23 the **System.Windows.Forms.ControlBindingsCollection** . By adding  
24 **System.Windows.Forms.Binding** objects to the collection, you can bind any  
25 property of a control to the property of an object.

*mmm) DefaultBackColor*

*nnn) ToString*

```
[C#]      public      static      Color      DefaultBackColor      {get;}
[C++]     public:     __property static      Color      get_DefaultBackColor();
[VB]      Public     Shared     ReadOnly     Property     DefaultBackColor     As     Color
[JScript] public     static     function     get     DefaultBackColor()     :     Color;
```

*Description*

Gets the default background color of the control.

This is the default **System.Windows.Forms.Control.BackColor** property value of a generic top-level control. Derived classes may have different defaults.

*ooo) DefaultFont*

*ppp) ToString*

```
[C#]      public      static      Font      DefaultFont      {get;}
[C++]     public:     __property static      Font*      get_DefaultFont();
[VB]      Public     Shared     ReadOnly     Property     DefaultFont     As     Font
[JScript] public     static     function     get     DefaultFont()     :     Font;
```

*Description*

Gets the default font of the control.

If no **System.Drawing.FontFamily.GenericSansSerif** fonts are installed on the user's computer, the DEFAULT\_GUI\_FONT is used. The DEFAULT\_GUI\_FONT is the default font used by user interface objects such as menus and dialog boxes.

1        *qqq) DefaultForeColor*

2        *rrr) ToString*

3  
4 [C#]        public        static        Color        DefaultForeColor        {get;}  
5 [C++]        public:        \_\_property        static        Color        get\_DefaultForeColor();  
6 [VB]        Public        Shared        ReadOnly        Property        DefaultForeColor        As        Color  
7 [JScript]        public        static        function        get        DefaultForeColor()        :        Color;

8  
9        *Description*

10        Gets the default foreground color of the control.

11        This is the default **System.Windows.Forms.Control.ForeColor** property  
12        value of a non-parented control. Derived classes may have different defaults.

13        *sss) DefaultImeMode*

14        *ttt) ToString*

15 [C#]        protected        virtual        ImeMode        DefaultImeMode        {get;}  
16 [C++]        protected:        \_\_property        virtual        ImeMode        get\_DefaultImeMode();  
17 [VB]        Overridable        Protected        ReadOnly        Property        DefaultImeMode        As        ImeMode  
18 [JScript]        protected        function        get        DefaultImeMode()        :        ImeMode;

19  
20        *Description*

21        Gets the default Input Method Editor (IME) mode supported by this control.

22        An input method editor (IME) is a program that allows users to enter complex  
23        characters and symbols, such as Japanese Kanji characters, by using a standard  
24        keyboard.

uuu) *DefaultSize*

vvv) *ToString*

```
[C#]      protected      virtual      Size      DefaultSize      {get;}
[C++]     protected:     __property  virtual      Size      get_DefaultSize();
[VB]      Overridable    Protected    ReadOnly    Property    DefaultSize    As    Size
[JScript]  protected     function    get         DefaultSize()    :    Size;
```

#### *Description*

Gets the default size of the control.

The **System.Windows.Forms.Control.DefaultSize** property represents the **System.Drawing.Size** of the control when it is initially created. You may adjust the size of the control by setting its **System.Windows.Forms.Control.Size** property value.

www) *DesignMode*

xxx) *DisplayRectangle*

yyy) *ToString*

#### *Description*

Gets the rectangle that represents the display area of the control.

The **System.Windows.Forms.Control.DisplayRectangle** property returns the client rectangle of the display area of the control. For the base control class, this is equal to the client rectangle. However, inheriting controls may want to change this if their client area differs from their display area. The display rectangle is the smallest **System.Drawing.Rectangle** that encloses a control and is used used to layout controls. If When overriding the **System.Windows.Forms.Control.DisplayRectangle** property in a derived class, use the base class's

**System.Windows.Forms.Control.DisplayRectangle** property to extend the base implementation. Alternatively, you must provide all the implementation.

*zzz) Disposing*

*aaaa) ToString*

[C#]	public	bool	Disposing	{get;}
[C++]	public:	__property	bool	get_Disposing();
[VB]	Public	ReadOnly	Property	Disposing As Boolean
[JScript]	public	function	get Disposing()	: Boolean;

### *Description*

Gets a value indicating whether the control is in the process of being disposed.

When this property returns **true**, the control is in the process of being disposed. After the control is disposed, it can no longer be referenced as a valid Windows control. Even though the instance of a control is disposed, it is still maintained in memory until it is removed from memory through garbage collection. When a control is disposed, you can not call its

**System.Windows.Forms.Control.RecreateHandle** method.

*bbbb) Dock*

*cccc) ToString*

[C#]	public	virtual	DockStyle	Dock	{get; set;}
[C++]	public:	__property	virtual	DockStyle	get_Dock();public: __property virtual
	void				set_Dock(DockStyle);
[VB]	Overridable	Public	Property	Dock	As DockStyle
[JScript]	public	function	get Dock()	: DockStyle;	public function set
			Dock(DockStyle);		



## Description

Gets or sets which edge of the parent container a control is docked to.

A control can be docked to one edge of its parent container or can dock to all edges and fill the parent container. For example, if you set this property to **System.Windows.Forms.DockStyle.Left**, the left edge of the control will be docked to the left edge of its parent control. Additionally, the docked edge of the control is resized to match that of its container control. Controls are docked in order of their z-order.

*dddd) Enabled*

*eeee) ToString*

```
[C#]          public          bool          Enabled          {get;          set;}
[C++] public:  __property  bool  get_Enabled();public:  __property  void
set_Enabled(bool);
[VB]          Public          Property          Enabled          As          Boolean
[JScript] public function get Enabled() : Boolean;public function set
Enabled(Boolean);
```

## Description

Gets or sets a value indicating whether the control can respond to user interaction.

The **System.Windows.Forms.Control.Enabled** property allows controls to be enabled or disabled at run time. For example, you can disable controls that do not apply to the current state of the application. You can also disable a control to restrict its use. For example, a button can be disabled to prevent the user from clicking it. If a control is disabled, it cannot be selected.

*ffff) Events*

*gggg) Focused*

*hhhh) ToString*

#### *Description*

Gets a value indicating whether the control has input focus.

When overriding the **System.Windows.Forms.Control.Focused** property in a derived class, use the base class's **System.Windows.Forms.Control.Focused** property to extend the base implementation. Otherwise, you must provide all the implementation.

*iiii) Font*

*jjjj) ToString*

[C#]            public            virtual            Font            Font            {get;            set;}

[C++] public: \_\_property virtual Font\* get\_Font();public: \_\_property virtual void  
set\_Font(Font\*);

[VB]            Overridable            Public            Property            Font            As            Font

[JScript] public function get Font() : Font;public function set Font(Font);

#### *Description*

Gets or sets the font of the text displayed by the control.

Because the **System.Drawing.Font** object is immutable (meaning that you cannot adjust any of it's properties), you can only assign the **System.Windows.Forms.Control.Font** property a new **System.Drawing.Font** object. However, you can base the new font on the existing font. The following is an example of how to adjust the existing font to make it bold: `myControl.Font = new Font(myControl.Font, myControl.Font.Style | FontStyle.Bold); MyControl.Font = New Font(MyControl.Font, _`

MyControl.Font.Style Or FontStyle.Bold) When overriding the **System.Windows.Forms.Control.Font** property in a derived class, use the base class's **System.Windows.Forms.Control.Font** property to extend the base implementation. Otherwise, you must provide all the implementation. You are not required to override both the **get** and **set** methods of the **System.Windows.Forms.Control.Font** property; you can override only one if needed.

*kkkk) FontHeight*

*llll) ToString*

[C#]           protected           int           FontHeight           {get;           set;}

[C++]   protected: \_\_property int get\_FontHeight();protected: \_\_property void set\_FontHeight(int);

[VB]           Protected           Property           FontHeight           As           Integer

[JScript]   protected   function   get   FontHeight() : int;protected   function   set   FontHeight(int);

### *Description*

Gets or sets the height of the font of the control.

The **System.Windows.Forms.Control.FontHeight** property should not be set to any value other than the control's **System.Drawing.Font.Height** value, or -1. Setting **System.Windows.Forms.Control.FontHeight** to -1 has the effect of clearing the cached height value, and the value is recalculated the next time the property is referenced.

*mmmm)ForeColor*

*nnnn) ToString*

[C#]           public           virtual           Color           ForeColor           {get;           set;}

[C++]   public: \_\_property virtual Color get\_ForeColor();public: \_\_property virtual

```

1 void set_ForeColor(Color);
2 [VB] Overridable Public Property ForeColor As Color
3 [JScript] public function get ForeColor() : Color;public function set
4 ForeColor(Color);

```

### Description

Gets or sets the foreground color of the control.

When overriding the **System.Windows.Forms.Control.ForeColor** property in a derived class, use the base class's **System.Windows.Forms.Control.ForeColor** property to extend to the base implementation. Otherwise, you must provide all the implementation. You are not required to override both the **get** and **set** methods of the **System.Windows.Forms.Control.ForeColor** property; you can override only one if needed.

*oooo) Handle*

*pppp) ToString*

```

15 [C#] public IntPtr Handle {get;}
16 [C++] public: __property IntPtr get_Handle();
17 [VB] Public ReadOnly Property Handle As IntPtr
18 [JScript] public function get Handle() : IntPtr;

```

### Description

Gets the window handle that the control is bound to.

The value of the **System.Windows.Forms.Control.Handle** property is a Windows HWND. If the handle has not yet been created, referencing this property will force the handle to be created.

*qqqq) HasChildren*

*rrrr) ToString*

```
[C#]          public          bool          HasChildren          {get;}
[C++]         public:         __property      bool          get_HasChildren();
[VB]         Public   ReadOnly   Property   HasChildren   As   Boolean
[JScript]     public   function   get       HasChildren()   :   Boolean;
```

#### *Description*

Gets a value indicating whether the control contains one or more child controls.

If the **System.Windows.Forms.Control.Controls** collection has a **System.Windows.Forms.Control.ControlCollection.Count** greater than zero, the **System.Windows.Forms.Control.HasChildren** property will return **true**. Accessing the **System.Windows.Forms.Control.HasChildren** property does not force the creation of a **System.Windows.Forms.Control.ControlCollection** if the control has no children, so referencing this property can provide a performance benefit when walking a tree of controls.

*ssss) Height*

*tttt) ToString*

```
[C#]          public          int          Height          {get;          set;}
[C++]         public:         __property      int          get_Height();public: __property void set_Height(int);
[VB]         Public          Property      Height          As          Integer
[JScript]     public   function   get       Height()       :   int;public function set Height(int);
```

#### *Description*

Gets or sets the height of the control.

The minimum height for the derived control **System.Windows.Forms.Splitter** is one pixel. The default height for the **System.Windows.Forms.Splitter** control is three pixels. Setting the height of the **System.Windows.Forms.Splitter** control to a value less than one will reset the property value to the default height.

*uuuu) ImeMode*

*vvvv) ToString*

[C#]            public            ImeMode            ImeMode            {get;            set;}

[C++]    public:    \_\_property    ImeMode    get\_ImeMode();public:    \_\_property    void  
set\_ImeMode(ImeMode);

[VB]            Public            Property            ImeMode            As            ImeMode

[JScript]    public    function    get    ImeMode()    :    ImeMode;public    function    set  
ImeMode(ImeMode);

#### *Description*

Gets or sets the Input Method Editor (IME) mode of the control.

An input method editor (IME) is a program that allows users to enter complex characters and symbols, such as Japanese Kanji characters, using a standard keyboard. The **System.Windows.Forms.Control.ImeMode** property is typically set to **System.Windows.Forms.ImeMode.Off** for a **System.Windows.Forms.TextBox** that is intended to only enter numeric values.

*www)InvokeRequired*

*xxxx) ToString*

[C#]            public            bool            InvokeRequired            {get;}

[C++]            public:            \_\_property            bool            get\_InvokeRequired();

[VB]            Public            ReadOnly            Property            InvokeRequired            As            Boolean

[JScript] public function get InvokeRequired() : Boolean;

### Description

Gets a value indicating whether the caller must call invoke when making method calls to this control because the caller is on a different thread than the control was created on.

Controls in Windows Forms are bound to a specific thread and are not thread-safe. Therefore, if you are calling a control's method from a different thread, you must use the control's invoke method to marshal the call to the proper thread. This function can be used to determine if you must call invoke, which can be useful if you do not know what thread owns a control. There are four methods on a control that are safe to call from any thread:

**System.Windows.Forms.Control.Invoke(System.Delegate)** ,  
**System.Windows.Forms.Control.BeginInvoke(System.Delegate)** ,  
**System.Windows.Forms.Control.EndInvoke(System.IAsyncResult)** and  
**System.Windows.Forms.Control.CreateGraphics** . For all other method calls, you should use one of the invoke methods when calling from a different thread.

yyyy) *IsAccessible*

zzzz) *ToString*

[C#] public bool IsAccessible {get; set;}

[C++] public: \_\_property bool get\_IsAccessible();public: \_\_property void set\_IsAccessible(bool);

[VB] Public Property IsAccessible As Boolean

[JScript] public function get IsAccessible() : Boolean;public function set IsAccessible(Boolean);

### Description

Gets or sets a value indicating whether the control is visible to accessibility applications.

*aaaaa) IsDisposed*

*bbbbbb) ToString*

[C#]	public	bool	IsDisposed	{get;}
[C++]	public:	__property	bool	get_IsDisposed();
[VB]	Public	ReadOnly	Property	IsDisposed As Boolean
[JScript]	public	function	get	IsDisposed() : Boolean;

*Description*

Gets a value indicating whether the control has been disposed of.

When this property returns **true** , the control is disposed of and can no longer be referenced as a valid Windows control. Even though the instance of a control is disposed of, it is still maintained in memory until it is removed from memory through garbage collection. When a control is disposed, you can not call its **System.Windows.Forms.Control.RecreateHandle** method.

*cccccc) IsHandleCreated*

*dddddd) ToString*

[C#]	public	bool	IsHandleCreated	{get;}
[C++]	public:	__property	bool	get_IsHandleCreated();
[VB]	Public	ReadOnly	Property	IsHandleCreated As Boolean
[JScript]	public	function	get	IsHandleCreated() : Boolean;

*Description*

Gets a value indicating whether the control has a handle associated with it.



*eeeeee) Left*

*ffffff) ToString*

```
[C#]          public          int          Left          {get;          set;}
[C++] public: __property int get_Left();public: __property void set_Left(int);
[VB]          Public          Property          Left          As          Integer
[JScript] public function get Left() : int;public function set Left(int);
```

#### *Description*

Gets or sets the x-coordinate of a control's left edge in pixels.

The **System.Windows.Forms.Control.Left** property value is equivalent to the **System.Drawing.Point.X** property of the **System.Windows.Forms.Control.Location** property value of the control.

*ggggg) Location*

*hhhhh)ToString*

```
[C#]          public          Point          Location          {get;          set;}
[C++] public: __property Point get_Location();public: __property void
set_Location(Point);
[VB]          Public          Property          Location          As          Point
[JScript] public function get Location() : Point;public function set Location(Point);
```

#### *Description*

Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.

Since the **System.Drawing.Point** class is a value type (structure in VB, struct in C#) it is returned by value, meaning accessing the property returns a copy of

the upper-left point of the control. So, adjusting the **System.Drawing.Point.X** or **System.Drawing.Point.Y** properties of the **System.Drawing.Point** object returned from this property will not affect the **System.Windows.Forms.Control.Left** , **System.Windows.Forms.Control.Right** , **System.Windows.Forms.Control.Top** , or **System.Windows.Forms.Control.Bottom** property values of the control. To adjust these properties set each property value individually, or set the **System.Windows.Forms.Control.Location** property with a new **System.Drawing.Point** object.

*iiii) ModifierKeys*

*jjjj) ToString*

```
[C#]      public      static      Keys      ModifierKeys      {get;}
[C++]     public:     __property static      Keys      get_ModifierKeys();
[VB]      Public     Shared     ReadOnly     Property     ModifierKeys     As     Keys
[JScript] public     static     function     get     ModifierKeys()     :     Keys;
```

### *Description*

Gets a value indicating which of the modifier keys (SHIFT, CTRL, and ALT) is in a pressed state.

*kkkkk) MouseButtons*

*llll) ToString*

```
[C#]      public      static      MouseButtons      MouseButtons      {get;}
[C++]     public:     __property static      MouseButtons      get_MouseButtons();
[VB]      Public     Shared     ReadOnly     Property     MouseButtons     As     MouseButtons
[JScript] public     static     function     get     MouseButtons()     :     MouseButtons;
```

## Description

Gets a value indicating which of the mouse buttons is in a pressed state.

*mmmmm)MousePosition*

*nnnnn)ToString*

```
[C#]      public      static      Point      MousePosition      {get;}
[C++]     public:     __property static Point get_MousePosition();
[VB]      Public      Shared      ReadOnly Property MousePosition As Point
[JScript] public static function get MousePosition() : Point;
```

## Description

Gets the position of the mouse cursor in screen coordinates.

The **System.Windows.Forms.Control.MousePosition** property returns a **System.Drawing.Point** that represents the mouse cursor position at the time the property was referenced.

*ooooo) Name*

*ppppp) ToString*

```
[C#]      public      string      Name      {get;      set;}
[C++]     public:     __property String* get_Name();public: __property void
set_Name(String*);
[VB]      Public      Property      Name      As      String
[JScript] public function get Name() : String;public function set Name(String);
```

## Description

Gets or sets the name of the control.

The **System.Windows.Forms.Control.Name** property can be used at run time to evaluate the object by name rather than type and programmatic name. Because the **System.Windows.Forms.Control.Name** property returns a **System.String** type, it can be evaluated in case-style logic statements ( **Select** statement in Visual Basic, **switch** statement in C# and C++).

*qqqqq) Parent*

*rrrrr) ToString*

[C#]            public            Control            Parent            {get;            set;}

[C++]   public:   \_\_property   Control\*   get\_Parent();public:   \_\_property   void  
set\_Parent(Control\*);

[VB]            Public            Property            Parent            As            Control

[JScript] public function get Parent() : Control;public function set Parent(Control);

## Description

Gets or sets the parent container of the control.

Setting the **System.Windows.Forms.Control.Parent** property value to **null** removes this control from the **System.Windows.Forms.Control.ControlCollection** of its current parent control.

*sssss) ProductName*

*ttttt) ToString*

[C#]            public            string            ProductName            {get;}

[C++]   public:            \_\_property            String\*            get\_ProductName();

```

1 [VB]      Public      ReadOnly      Property      ProductName      As      String
2 [JScript]      public      function      get      ProductName()      :      String;

```

#### *Description*

Gets the product name of the assembly containing the control.

The **System.Windows.Forms.Control.ProductName** property is a read-only property. In order to change the value of this property, set the **System.Reflection.AssemblyProductAttribute.Product** property value of the **System.Reflection.AssemblyProductAttribute** .

*uuuuu)ProductVersion*

*vvvvv) ToString*

```

11 [C#]      public      string      ProductVersion      {get;}
12 [C++]      public:      __property      String*      get_ProductVersion();
13 [VB]      Public      ReadOnly      Property      ProductVersion      As      String
14 [JScript]      public      function      get      ProductVersion()      :      String;

```

#### *Description*

Gets the version of the assembly containing the control.

The **System.Windows.Forms.Control.ProductVersion** property is a read-only property. In order to change the value of this property, set the **System.Reflection.AssemblyVersionAttribute.Version** property value of the **System.Reflection.AssemblyVersionAttribute** .

*wwwww)RecreatingHandle*

*xxxxx) ToString*

```

24 [C#]      public      bool      RecreatingHandle      {get;}

```

```

1 [C++]      public:      __property      bool      get_RecreatingHandle();
2 [VB]      Public      ReadOnly      Property      RecreatingHandle      As      Boolean
3 [JScript]      public      function      get      RecreatingHandle()      :      Boolean;

```

#### *Description*

Gets a value indicating whether the control is currently re-creating its handle.

*yyyyy) Region*

*zzzzz) ToString*

```

10 [C#]      public      Region      Region      {get;      set;}
11 [C++]      public:      __property      Region*      get_Region();public:      __property      void
12      set_Region(Region*);
13 [VB]      Public      Property      Region      As      Region
14 [JScript]      public      function      get      Region()      :      Region;public      function      set
15      Region(Region);

```

#### *Description*

Gets or sets the window region associated with the control.

The window region is an elliptical or polygonal area within the window where the operating system permits drawing. The operating system does not display any portion of a window that lies outside of the window region. The coordinates of a control's region are relative to the upper-left corner of the control, not the client area of the control.

*aaaaaa)RenderRightToLeft*

*bbbbbb)ToString*

```
[C#]          protected          bool          RenderRightToLeft          {get;}
[C++]          protected:          __property          bool          get_RenderRightToLeft();
[VB]          Protected          ReadOnly          Property          RenderRightToLeft          As          Boolean
[JScript] protected function get RenderRightToLeft() : Boolean;
```

*ccccc)ResizeRedraw*

*dddddd)ToString*

```
[C#]          protected          bool          ResizeRedraw          {get;          set;}
[C++] protected: __property bool get_ResizeRedraw();protected: __property void
set_ResizeRedraw(bool);
[VB]          Protected          Property          ResizeRedraw          As          Boolean
[JScript] protected function get ResizeRedraw() : Boolean;protected function set
ResizeRedraw(Boolean);
```

### *Description*

Gets or sets a value indicating whether the control redraws itself when resized.

The **System.Windows.Forms.Control.ResizeRedraw** property value is equivalent to the return value of the **System.Windows.Forms.Control.GetStyle(System.Windows.Forms.ControlStyles)** method when passing in the **System.Windows.Forms.ControlStyles.ResizeRedraw** value as a parameter.

*eeeeee)Right*

*ffffff) ToString*

[C#]	public	int	Right	{get;}
[C++]	public:	__property	int	get_Right();
[VB]	Public	ReadOnly	Property	Right As Integer
[JScript]	public	function	get	Right() : int;

*Description*

Gets the distance between the right edge of the control and the left edge of its container.

The value of the **System.Windows.Forms.Control.Right** property is equal to the sum of the **System.Windows.Forms.Control.Left** property value and the **System.Windows.Forms.Control.Width** property value.

*gggggg)RightToLeft*

*hhhhh)ToString*

[C#]	public	virtual	RightToLeft	RightToLeft	{get; set;}
[C++]	public:	__property	virtual	RightToLeft	get_RightToLeft();public:
	__property	virtual	void	set_RightToLeft(RightToLeft);	
[VB]	Overridable	Public	Property	RightToLeft	As RightToLeft
[JScript]	public	function	get	RightToLeft()	: RightToLeft;public function set
				RightToLeft(RightToLeft);	

*Description*

Gets or sets a value indicating whether control's elements are aligned to support locales using right-to-left fonts.



The **System.Windows.Forms.Control.RightToLeft** property is used for international applications where the language is written from right to left, such as Hebrew or Arabic. When this property is set to **System.Windows.Forms.RightToLeft.Yes**, control elements that include text are displayed from right to left.

*iiiiii) ShowFocusCues*

*jjjjj) ToString*

```
[C#]      protected      virtual      bool      ShowFocusCues      {get;}
[C++]      protected:      __property      virtual      bool      get_ShowFocusCues();
[VB]      Overridable Protected ReadOnly Property ShowFocusCues As Boolean
[JScript]      protected      function      get      ShowFocusCues()      :      Boolean;
```

#### *Description*

Gets a value indicating whether the control should display focus rectangles.

For more information on this feature see the **WM\_CHANGEUISTATE**, **WM\_QUERYUISTATE**, and **WM\_UPDATEUISTATE** topics located in the Windows Platform SDK in the MSDN Library.

*kkkkkk)ShowKeyboardCues*

*lllll) ToString*

```
[C#]      protected      bool      ShowKeyboardCues      {get;}
[C++]      protected:      __property      bool      get_ShowKeyboardCues();
[VB]      Protected ReadOnly Property ShowKeyboardCues As Boolean
[JScript]      protected      function      get      ShowKeyboardCues()      :      Boolean;
```

#### *Description*

Gets a value indicating whether the control should display keyboard shortcuts.

For more information on this feature see the **WM\_CHANGEUISTATE** ,  
**WM\_QUERYUISTATE** , and **WM\_UPDATEUISTATE** topics located in the  
Windows Platform SDK in the MSDN Library.

*mmmmm)Site*

*nnnnn)ToString*

[C#]        public        override        ISite        Site        {get;        set;}

[C++] public: \_\_property virtual ISite\* get\_Site();public: \_\_property virtual void  
set\_Site(ISite\*);

[VB]        Overrides        Public        Property        Site        As        ISite

[JScript] public function get Site() : ISite;public function set Site(ISite);

#### *Description*

Gets or sets the site of the control.

*ooooo)Size*

*ppppp)ToString*

[C#]        public        Size        Size        {get;        set;}

[C++] public: \_\_property Size get\_Size();public: \_\_property void set\_Size(Size);

[VB]        Public        Property        Size        As        Size

[JScript] public function get Size() : Size;public function set Size(Size);

#### *Description*

Gets or sets the height and width of the control.

Since the **System.Drawing.Size** class is a value type (structure in VB, struct in C#) it is returned by value, meaning accessing the property returns a copy of the size of the control. So, adjusting the **System.Drawing.Size.Width** or

**System.Drawing.Size.Height** properties of the **System.Drawing.Size** object returned from this property will not affect the **System.Windows.Forms.Control.Width** and **System.Windows.Forms.Control.Height** of the control. To adjust the **System.Windows.Forms.Control.Width** or **System.Windows.Forms.Control.Height** of the control you must use the **System.Windows.Forms.Control.Width** or **System.Windows.Forms.Control.Height** property, or set the **System.Windows.Forms.Control.Size** property with a new **System.Drawing.Size** object.

*qqqqqq) TabIndex*

*rrrrrr) ToString*

[C#]            public            int            TabIndex            {get;            set;}

[C++]   public:   \_\_property   int   get\_TabIndex();public:   \_\_property   void  
set\_TabIndex(int);

[VB]            Public            Property            TabIndex            As            Integer

[JScript] public function get TabIndex() : int;public function set TabIndex(int);

### *Description*

Gets or sets the tab order of the control within its container.

A tab index may consist of any valid integer greater than or equal to zero, lower numbers being earlier in the tab order. If more than one control on the same parent control has the same tab index, the z-order of the controls determines the order to cycle through the controls.

*ssssss) TabStop*

*tttttt) ToString*

[C#]            public            bool            TabStop            {get;            set;}

[C++]   public:   \_\_property   bool   get\_TabStop();public:   \_\_property   void

1 set\_TabStop(bool);

2 [VB] Public Property TabStop As Boolean

3 [JScript] public function get TabStop() : Boolean;public function set

4 TabStop(Boolean);

6 *Description*

7 Gets or sets a value indicating whether the user can give the focus to this control  
using the TAB key.

8 When the user presses the TAB key, the input focus is set to the next control in  
9 the tab order. Controls with the **System.Windows.Forms.Control.TabStop**  
property value of **false** are not included in the collection of controls in the tab  
10 order. The tab order can be manipulated by setting the control's  
**System.Windows.Forms.Control.TabIndex** property value.

11 *uuuuuu)Tag*

12 *vvvvvv)ToString*

13  
14  
15 [C#] public object Tag {get; set;}

16 [C++] public: \_\_property Object\* get\_Tag();public: \_\_property void

17 set\_Tag(Object\*);

18 [VB] Public Property Tag As Object

19 [JScript] public function get Tag() : Object;public function set Tag(Object);

21 *Description*

22 Gets or sets the object that contains data about the control.

23 Any type derived from the **System.Object** class can be assigned to this  
property. If the **System.Windows.Forms.Control.Tag** property is set through  
24 the Windows Forms designer, only text may be assigned.

*wwwwww)Text*

*xxxxxx)ToString*

```
[C#]      public      virtual      string      Text      {get;      set;}
[C++] public: __property virtual String* get_Text();public: __property virtual
void                                             set_Text(String*);
[VB]      Overridable      Public      Property      Text      As      String
[JScript] public function get Text() : String;public function set Text(String);
```

### *Description*

Gets or sets the text associated with this control.

The **System.Windows.Forms.Control.Text** property of the control is used differently by each derived class. For example the **System.Windows.Forms.Control.Text** property of a **System.Windows.Forms.Form** is displayed in the title bar at the top of the form, is fairly small in character count, and usually displays the application or document name. However, the **System.Windows.Forms.Control.Text** property of a **System.Windows.Forms.RichTextBox** can be great in size and can include numerous nonvisual characters used to format the text. For example, the text displayed in a **System.Windows.Forms.RichTextBox** can be formatted by adjusting the **System.Drawing.Font** properties, or by the addition of spaces or tab characters to align the text.

*yyyyyy)Top*

*zzzzzz) ToString*

```
[C#]      public      int      Top      {get;      set;}
[C++] public: __property int get_Top();public: __property void set_Top(int);
[VB]      Public      Property      Top      As      Integer
[JScript] public function get Top() : int;public function set Top(int);
```

## Description

Gets or sets the y-coordinate of a control's top edge in pixels.

The **System.Windows.Forms.Control.Top** property value is equivalent to the **System.Drawing.Point.Y** property of the **System.Windows.Forms.Control.Location** property value of the control.

*aaaaaaa)TopLevelControl*

*bbbbbbb)ToString*

```
[C#]          public          Control          TopLevelControl          {get;}
[C++]         public:         __property      Control*          get_TopLevelControl();
[VB]          Public          ReadOnly          Property          TopLevelControl          As          Control
[JScript]     public          function          get          TopLevelControl()          :          Control;
```

## Description

Gets the control that is not parented to another Windows Form control. Typically, this is the outermost **System.Windows.Forms.Form** that the control is contained in.

The top-level control is defined as the control that is not parented to another Windows Form control. Typically, this is the outermost **System.Windows.Forms.Form** that the control is contained in. For example, if the control is contained on an MDI child **System.Windows.Forms.Form** , then the top-level control is the MDI parent **System.Windows.Forms.Form** .

*ccccccc)Visible*

*ddddddd)ToString*

```
[C#]          public          bool          Visible          {get;          set;}
[C++]         public:         __property      bool          get_Visible();public:         __property      void
```

```

1 set_Visible(bool);
2 [VB]      Public      Property      Visible      As      Boolean
3 [JScript] public function get Visible() : Boolean;public function set
4 Visible(Boolean);
5

```

#### *Description*

Gets or sets a value indicating whether the control is displayed.

*eeeeeee)Width*

*ffffff) ToString*

```

11 [C#]      public      int      Width      {get;      set;}
12 [C++] public: __property int get_Width();public: __property void set_Width(int);
13 [VB]      Public      Property      Width      As      Integer
14 [JScript] public function get Width() : int;public function set Width(int);
15

```

#### *Description*

Gets or sets the width of the control.

*ggggggg)WindowTarget*

*hhhhhhh)ToString*

```

21 [C#]      public      IWindowTarget      WindowTarget      {get;      set;}
22 [C++] public: __property IWindowTarget* get_WindowTarget();public:
23 __property      void      set_WindowTarget(IWindowTarget*);
24 [VB]      Public      Property      WindowTarget      As      IWindowTarget
25

```

[JScript] public function get WindowTarget() : IWindowTarget; public function set WindowTarget(IWindowTarget);

#### *Description*

The target of Win32 window messages.

#### *iiiiii) ToString*

[C#]	public	event	EventHandler	BackColorChanged;
[C++]	public:	__event	EventHandler*	BackColorChanged;
[VB]	Public	Event	BackColorChanged	As EventHandler

#### *Description*

Occurs when the value of the **System.Windows.Forms.Control.BackColor** property changes.

This event will be raised if the **System.Windows.Forms.Control.BackColor** property is changed either through a programmatic change or through runtime user interaction.

#### *jjjjjj) ToString*

[C#]	public	event	EventHandler	BackgroundImageChanged;
[C++]	public:	__event	EventHandler*	BackgroundImageChanged;
[VB]	Public	Event	BackgroundImageChanged	As EventHandler

#### *Description*

Occurs when the value of the **System.Windows.Forms.Control.BackgroundImage** property changes.



This event will be raised if the **System.Windows.Forms.Control.BackgroundImage** property is changed either through a programmatic change or through runtime user interaction.

*kkkkkkk)ToString*

[C#]	public	event	EventHandler	BindingContextChanged;
[C++]	public:	__event	EventHandler*	BindingContextChanged;
[VB]	Public	Event	BindingContextChanged	As EventHandler

*Description*

Occurs when the value of the **System.Windows.Forms.BindingContext** property changes.

To add a new **System.Windows.Forms.BindingContext** to the **System.Windows.Forms.Control** through the **System.Windows.Forms.Control.BindingContext** property, see the **System.Windows.Forms.BindingContext.#ctor** constructor.

*lllllll) ToString*

[C#]	public	event	EventHandler	CausesValidationChanged;
[C++]	public:	__event	EventHandler*	CausesValidationChanged;
[VB]	Public	Event	CausesValidationChanged	As EventHandler

*Description*

Occurs when the value of the **System.Windows.Forms.Control.CausesValidation** property changes.

This event will be raised if the **System.Windows.Forms.Control.CausesValidation** property is changed either through a programmatic change or through runtime user interaction.

### *mmmmmmm)ToString*

```
1  
2 [C#]      public      event      UICuesEventHandler      ChangeUICues;  
3 [C++]     public:     __event     UICuesEventHandler*      ChangeUICues;  
4 [VB]      Public      Event      ChangeUICues      As      UICuesEventHandler  
5  
6
```

#### *Description*

Occurs when the focus or keyboard user interface (UI) cues change.

For more information about handling events, see .

### *nnnnnnn)ToString*

```
11 [C#]      public      event      EventHandler      Click;  
12 [C++]     public:     __event     EventHandler*      Click;  
13 [VB]      Public      Event      Click      As      EventHandler  
14  
15
```

#### *Description*

Occurs when the control is clicked.

The **System.Windows.Forms.ControlStyles.StandardClick** style must be set for this event to be raised.

### *ooooooo)ToString*

```
21 [C#]      public      event      EventHandler      ContextMenuChanged;  
22 [C++]     public:     __event     EventHandler*      ContextMenuChanged;  
23 [VB]      Public      Event      ContextMenuChanged      As      EventHandler  
24  
25
```

#### *Description*

Occurs when the value of the  
**System.Windows.Forms.Control.ContextMenu** property changes.

This event will be raised if the  
**System.Windows.Forms.Control.ContextMenu** property is changed either  
through a programmatic change or through runtime user interaction.

*ppppppp)ToString*

[C#]	public	event	ControlEventHandler	ControlAdded;
[C++]	public:	__event	ControlEventHandler*	ControlAdded;
[VB]	Public	Event	ControlAdded	As ControlEventHandler

#### *Description*

Occurs when a new control is added to the  
**System.Windows.Forms.Control.ControlCollection** .

For more information about handling events, see .

*qqqqqqq)ToString*

[C#]	public	event	ControlEventHandler	ControlRemoved;
[C++]	public:	__event	ControlEventHandler*	ControlRemoved;
[VB]	Public	Event	ControlRemoved	As ControlEventHandler

#### *Description*

Occurs when a control is removed from the  
**System.Windows.Forms.Control.ControlCollection** .

For more information about handling events, see .

*rrrrrr)ToString*

[C#]	public	event	EventHandler	CursorChanged;
[C++]	public:	__event	EventHandler*	CursorChanged;
[VB]	Public	Event	CursorChanged	As EventHandler

*Description*

Occurs when the value of the **System.Windows.Forms.Control.Cursor** property changes.

This event will be raised if the **System.Windows.Forms.Control.Cursor** property is changed either through a programmatic changed or through runtime user interaction.

*ssssss)ToString*

*Description*

Occurs when the value of the **System.Windows.Forms.Control.Dock** property changes.

This event will be raised if the **System.Windows.Forms.Control.Dock** property is changed either through a programmatic changed or through runtime user interaction.

*tttttt) ToString*

[C#]	public	event	EventHandler	DoubleClick;
[C++]	public:	__event	EventHandler*	DoubleClick;
[VB]	Public	Event	DoubleClick	As EventHandler

*Description*

Occurs when the control is double-clicked.

The **System.Windows.Forms.ControlStyles.StandardClick** style must be set for this event to be raised.

*uuuuuuu)ToString*

[C#]	public	event	DragEventHandler	DragDrop;
[C++]	public:	__event	DragEventHandler*	DragDrop;
[VB]	Public	Event	DragDrop	As DragEventHandler

#### *Description*

Occurs when a drag-and-drop operation is completed.

For more information about handling events, see .

*vvvvvvv)ToString*

[C#]	public	event	DragEventHandler	DragEnter;
[C++]	public:	__event	DragEventHandler*	DragEnter;
[VB]	Public	Event	DragEnter	As DragEventHandler

#### *Description*

Occurs when an object is dragged into the control's bounds.

*wwwwwww)ToString*

[C#]	public	event	EventHandler	DragLeave;
[C++]	public:	__event	EventHandler*	DragLeave;
[VB]	Public	Event	DragLeave	As EventHandler

*Description*

Occurs when an object is dragged out of the control's bounds.

*xxxxxxx)ToString*

[C#]	public	event	DragEventHandler	DragOver;
[C++]	public:	__event	DragEventHandler*	DragOver;
[VB]	Public	Event	DragOver	As DragEventHandler

*Description*

Occurs when an object is dragged over the control's bounds.

*yyyyyyy)ToString*

[C#]	public	event	EventHandler	EnabledChanged;
[C++]	public:	__event	EventHandler*	EnabledChanged;
[VB]	Public	Event	EnabledChanged	As EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.Enabled** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Enabled** property is changed either through a programmatic changed or through runtime user interaction.

*zzzzzzzz)ToString*

[C#]	public	event	EventHandler	Enter;
------	--------	-------	--------------	--------

[C++]            public:            \_\_event            EventHandler\*            Enter;

[VB]            Public            Event            Enter            As            EventHandler

*Description*

Occurs when the control is entered.

For more information about handling events, see .

*aaaaaaaa)ToString*

[C#]            public            event            EventHandler            FontChanged;

[C++]            public:            \_\_event            EventHandler\*            FontChanged;

[VB]            Public            Event            FontChanged            As            EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.Font** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Font** property is changed either through a programmatic changed or through runtime user interaction.

*bbbbbbbb)ToString*

[C#]            public            event            EventHandler            ForeColorChanged;

[C++]            public:            \_\_event            EventHandler\*            ForeColorChanged;

[VB]            Public            Event            ForeColorChanged            As            EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.ForeColor** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.ForeColor** property is changed either through a programmatic changed or through runtime user interaction.

*cccccccc)ToString*

[C#]      public      event      GiveFeedbackEventHandler      GiveFeedback;

[C++]      public:      \_\_event      GiveFeedbackEventHandler\*      GiveFeedback;

[VB]      Public      Event      GiveFeedback      As      GiveFeedbackEventHandler

#### *Description*

Occurs during a drag operation.

The **System.Windows.Forms.Control.GiveFeedback** event allows the source of a drag event to modify the appearance of the mouse pointer in order to give the user visual feedback during a drag-and-drop operation.

*dddddddd)ToString*

[C#]              public              event              EventHandler              GotFocus;

[C++]              public:              \_\_event              EventHandler\*              GotFocus;

[VB]              Public              Event              GotFocus              As              EventHandler

#### *Description*

Occurs when the control receives focus.

Focus events occur in the following order:

**System.Windows.Forms.Control.EnterSystem.Windows.Forms.Control.GotFocusSystem.Windows.Forms.Control.LeaveSystem.Windows.Forms.Control.ValidatingSystem.Windows.Forms.Control.ValidatedSystem.Windows.Forms.Control.LostFocus** If the **System.Windows.Forms.Control.CausesValidation** property is set to **false**, the **System.Windows.Forms.Control.Validating** and **System.Windows.Forms.Control.Validated** events are suppressed.



eeeeeeee)ToString

[C#]	public	event	EventHandler	HandleCreated;
[C++]	public:	__event	EventHandler*	HandleCreated;
[VB]	Public	Event	HandleCreated	As EventHandler

Description

Occurs when a handle is created for the control.

For more information about handling events, see .

ffffff)ToString

[C#]	public	event	EventHandler	HandleDestroyed;
[C++]	public:	__event	EventHandler*	HandleDestroyed;
[VB]	Public	Event	HandleDestroyed	As EventHandler

Description

Occurs when the control's handle is in the process of being destroyed.

For more information about handling events, see .

gggggggg)ToString

[C#]	public	event	HelpEventHandler	HelpRequested;
[C++]	public:	__event	HelpEventHandler*	HelpRequested;
[VB]	Public	Event	HelpRequested	As HelpEventHandler

Description

Occurs when the user requests help for a control.

For more information about handling events, see .

*hhhhhhh)ToString*

[C#]	public	event	EventHandler	ImeModeChanged;
[C++]	public:	__event	EventHandler*	ImeModeChanged;
[VB]	Public	Event	ImeModeChanged	As EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.ImeMode** property has changed.

This event will be raised if the **System.Windows.Forms.Control.ImeMode** property is changed either through a programmatic changed or through runtime user interaction.

*iiiiiii) ToString*

[C#]	public	event	InvalidateEventHandler	Invalidated;
[C++]	public:	__event	InvalidateEventHandler*	Invalidated;
[VB]	Public	Event	Invalidated	As InvalidateEventHandler

*Description*

Occurs when a control's display requires redrawing.

For more information about handling events, see .

*jjjjjjj) ToString*

[C#]	public	event	KeyEventHandler	KeyDown;
------	--------	-------	-----------------	----------

1 [C++] public: \_\_event KeyEventHandler\* KeyDown;

2 [VB] Public Event KeyDown As KeyEventHandler

3  
4 *Description*

5 Occurs when a key is pressed while the control has focus.

6 For more information about handling events, see .

7 **kkkkkkkk) ToString**

8  
9 [C#] public event KeyPressEventHandler KeyPress;

10 [C++] public: \_\_event KeyPressEventHandler\* KeyPress;

11 [VB] Public Event KeyPress As KeyPressEventHandler

12  
13 *Description*

14 Occurs when a key is pressed while the control has focus.

15 For more information about handling events, see .

16 **lllllll) ToString**

17  
18 [C#] public event KeyEventHandler KeyUp;

19 [C++] public: \_\_event KeyEventHandler\* KeyUp;

20 [VB] Public Event KeyUp As KeyEventHandler

21  
22 *Description*

23 Occurs when a key is released while the control has focus.

24 For more information about handling events, see .

*mmmmmmmm)ToString*

[C#]	public	event	LayoutEventHandler	Layout;
[C++]	public:	__event	LayoutEventHandler*	Layout;
[VB]	Public	Event	Layout As	LayoutEventHandler

*Description*

Occurs when a control should reposition its child controls.

The **System.Windows.Forms.Control.Layout** event occurs when child controls are added or removed, when the bounds of the control changes, and when other changes occur that may affect the layout of the control. The layout event can be suppressed using the **System.Windows.Forms.Control.SuspendLayout** and **System.Windows.Forms.Control.ResumeLayout** methods. Suspending layout allows for multiple actions to be performed on a control without having to perform a layout for each change. For example, if you resize and move a control, each operation would raise a **System.Windows.Forms.Control.Layout** event.

*nnnnnnnn)ToString*

[C#]	public	event	EventHandler	Leave;
[C++]	public:	__event	EventHandler*	Leave;
[VB]	Public	Event	Leave As	EventHandler

*Description*

Occurs when the control is left.

For more information about handling events, see .

*oooooooo)ToString*

[C#]	public	event	EventHandler	LocationChanged;
------	--------	-------	--------------	------------------

[C++] public: \_\_event EventHandler\* LocationChanged;

[VB] Public Event LocationChanged As EventHandler

#### Description

Occurs when the **System.Windows.Forms.Control.Location** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Location** property is changed either through a programmatic changed or through runtime user interaction.

*pppppppp)ToString*

[C#] public event EventHandler LostFocus;

[C++] public: \_\_event EventHandler\* LostFocus;

[VB] Public Event LostFocus As EventHandler

#### Description

Occurs when the control loses focus.

Focus events occur in the following order:

**System.Windows.Forms.Control.EnterSystem.Windows.Forms.Control.GotFocusSystem.Windows.Forms.Control.LeaveSystem.Windows.Forms.Control.ValidatingSystem.Windows.Forms.Control.ValidatedSystem.Windows.Forms.Control.LostFocus** If the **System.Windows.Forms.Control.CausesValidation** property is set to **false**, the **System.Windows.Forms.Control.Validating** and **System.Windows.Forms.Control.Validated** events are suppressed.

*qqqqqqqq)ToString*

[C#] public event MouseEventHandler MouseDown;

[C++] public: \_\_event MouseEventHandler\* MouseDown;

[VB]      Public      Event      MouseDown      As      MouseEventHandler

*Description*

Occurs when the mouse pointer is over the control and a mouse button is pressed.

For more information about handling events, see .

*rrrrrrrr)ToString*

[C#]            public            event            EventHandler            MouseEnter;

[C++]            public:            \_\_event            EventHandler\*            MouseEnter;

[VB]            Public            Event            MouseEnter            As            EventHandler

*Description*

Occurs when the mouse pointer enters the control.

For more information about handling events, see .

*ssssssss)ToString*

[C#]            public            event            EventHandler            MouseHover;

[C++]            public:            \_\_event            EventHandler\*            MouseHover;

[VB]            Public            Event            MouseHover            As            EventHandler

*Description*

Occurs when the mouse pointer hovers over the control.

For more information about handling events, see .

*ttttttt) ToString*

[C#]	public	event	EventHandler	MouseLeave;
[C++]	public:	__event	EventHandler*	MouseLeave;
[VB]	Public	Event	MouseLeave	As EventHandler

*Description*

Occurs when the mouse pointer leaves the control.

For more information about handling events, see .

*uuuuuuuu) ToString*

[C#]	public	event	MouseEventHandler	MouseMove;
[C++]	public:	__event	MouseEventHandler*	MouseMove;
[VB]	Public	Event	MouseMove	As MouseEventHandler

*Description*

Occurs when the mouse pointer is moved over the control.

For more information about handling events, see .

*vvvvvvvv) ToString*

[C#]	public	event	MouseEventHandler	MouseUp;
[C++]	public:	__event	MouseEventHandler*	MouseUp;
[VB]	Public	Event	MouseUp	As MouseEventHandler

*Description*

Occurs when the mouse pointer is over the control and a mouse button is released.

For more information about handling events, see .

*wwwwwwwww)ToString*

[C#]	public	event	MouseEventHandler	MouseWheel;
[C++]	public:	__event	MouseEventHandler*	MouseWheel;
[VB]	Public	Event	MouseWheel	As MouseEventHandler

### *Description*

Occurs when the mouse wheel moves while the control has focus.

When handling the **System.Windows.Forms.Control.MouseWheel** event it is important to follow the user interface (UI) standards associated with the mouse wheel. The **System.Windows.Forms.MouseEventArgs.Delta** property value indicates the amount the mouse wheel has been moved. The UI should scroll when the accumulated delta is plus or minus 120. The UI should scroll the number of logical lines returned by the **System.Windows.Forms.SystemInformation.MouseWheelScrollLines** property for every delta value reached. You can also scroll more smoothly in smaller than 120 unit increments, however the ratio should remain constant, that is **System.Windows.Forms.SystemInformation.MouseWheelScrollLines** lines scrolled per 120 delta units of wheel movement.

*xxxxxxx)ToString*

[C#]	public	event	EventHandler	Move;
[C++]	public:	__event	EventHandler*	Move;
[VB]	Public	Event	Move	As EventHandler

### *Description*

Occurs when the control is moved.



For more information about handling events, see .

*yyyyyyyy)ToString*

[C#]	public	event	PaintEventHandler	Paint;
[C++]	public:	__event	PaintEventHandler*	Paint;
[VB]	Public	Event	Paint	As PaintEventHandler

*Description*

Occurs when the control is redrawn.

For more information about handling events, see .

*zzzzzzzz)ToString*

[C#]	public	event	EventHandler	ParentChanged;
[C++]	public:	__event	EventHandler*	ParentChanged;
[VB]	Public	Event	ParentChanged	As EventHandler

*Description*

Occurs when the **System.Windows.Forms.Control.Parent** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Parent** property is changed either through a programmatic changed or through runtime user interaction.

*aaaaaaaaa)ToString*

[C#]	public	event	QueryAccessibilityHelpEventHandler	QueryAccessibilityHelp;
[C++]	public:	__event	QueryAccessibilityHelpEventHandler*	

QueryAccessibilityHelp;

[VB]            Public            Event            QueryAccessibilityHelp            As

QueryAccessibilityHelpEventHandler

#### *Description*

Occurs when **System.Windows.Forms.AccessibleObject** is providing help to accessibility applications.

For more information about handling events, see .

*bbbbbbbb)ToString*

[C#]    public    event    QueryContinueDragEventHandler    QueryContinueDrag;

[C++]    public:    \_\_event    QueryContinueDragEventHandler\*    QueryContinueDrag;

[VB]    Public    Event    QueryContinueDrag    As    QueryContinueDragEventHandler

#### *Description*

Occurs during a drag-and-drop operation and allows the drag source to determine whether the drag-and-drop operation should be canceled.

For more information about handling events, see .

*cccccccc)ToString*

[C#]            public            event            EventHandler            Resize;

[C++]            public:            \_\_event            EventHandler\*            Resize;

[VB]            Public            Event            Resize            As            EventHandler

#### *Description*

Occurs when the control is resized.

It is preferable to use the **System.Windows.Forms.Control.Layout** event to handle custom layouts. The **System.Windows.Forms.Control.Layout** event is raised in response to **System.Windows.Forms.Control.Resize** events, but also in other conditions when layout may need to be applied.

#### *dddddddd)ToString*

[C#]	public	event	EventHandler	RightToLeftChanged;
[C++]	public:	__event	EventHandler*	RightToLeftChanged;
[VB]	Public	Event	RightToLeftChanged	As EventHandler

#### *Description*

Occurs when the **System.Windows.Forms.Control.RightToLeft** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.RightToLeft** property is changed either through a programmatic changed or through runtime user interaction.

#### *eeeeeeee)ToString*

[C#]	public	event	EventHandler	SizeChanged;
[C++]	public:	__event	EventHandler*	SizeChanged;
[VB]	Public	Event	SizeChanged	As EventHandler

#### *Description*

Occurs when the **System.Windows.Forms.Control.Size** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Size** property is changed either through a programmatic changed or through runtime user interaction.

*ffffffff)ToString*

[C#]	public	event	EventHandler	StyleChanged;
[C++]	public:	__event	EventHandler*	StyleChanged;
[VB]	Public	Event	StyleChanged	As EventHandler

*Description*

Occurs when the control style has changed.

The **System.Windows.Forms.Control.StyleChanged** event occurs when **System.Windows.Forms.ControlStyles** flags have been added or changed.

*ggggggggg)ToString*

[C#]	public	event	EventHandler	SystemColorsChanged;
[C++]	public:	__event	EventHandler*	SystemColorsChanged;
[VB]	Public	Event	SystemColorsChanged	As EventHandler

*Description*

Occurs when the system colors have changed.

This event will be raised if the **System.Drawing.SystemColors** is changed either through a programmatic changed or through runtime user interaction.

*hhhhhhhhh)ToString*

[C#]	public	event	EventHandler	TabIndexChanged;
[C++]	public:	__event	EventHandler*	TabIndexChanged;
[VB]	Public	Event	TabIndexChanged	As EventHandler

### Description

Occurs when the **System.Windows.Forms.Control.TabIndex** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.TabIndex** property is changed either through a programmatic changed or through runtime user interaction.

### iiiii) ToString

[C#]	public	event	EventHandler	TabStopChanged;
[C++]	public:	__event	EventHandler*	TabStopChanged;
[VB]	Public	Event	TabStopChanged	As EventHandler

### Description

Occurs when the **System.Windows.Forms.Control.TabStop** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.TabStop** property is changed either through a programmatic changed or through runtime user interaction.

### jjjjj) ToString

[C#]	public	event	EventHandler	TextChanged;
[C++]	public:	__event	EventHandler*	TextChanged;
[VB]	Public	Event	TextChanged	As EventHandler

### Description

Occurs when the **System.Windows.Forms.Control.Text** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Text** property is changed either through a programmatic changed or through runtime user interaction.

**kkkkkkkkk)ToString**

[C#]	public	event	EventHandler	Validated;
[C++]	public:	__event	EventHandler*	Validated;
[VB]	Public	Event	Validated As	EventHandler

*Description*

Occurs when the control is done validating.

For more information about handling events, see .

**lllllll) ToString**

[C#]	public	event	CancelEventHandler	Validating;
[C++]	public:	__event	CancelEventHandler*	Validating;
[VB]	Public	Event	Validating As	CancelEventHandler

*Description*

Occurs when the control is validating.

For more information about handling events, see .

**mmmmmmmm)ToString**

[C#]	public	event	EventHandler	VisibleChanged;
[C++]	public:	__event	EventHandler*	VisibleChanged;
[VB]	Public	Event	VisibleChanged As	EventHandler

## Description

Occurs when the **System.Windows.Forms.Control.Visible** property value has changed.

This event will be raised if the **System.Windows.Forms.Control.Visible** property is changed either through a programmatic changed or through runtime user interaction.

## *nnnnnnnnnn)AccessibilityNotifyClients*

[C#] protected void AccessibilityNotifyClients(AccessibleEvents accEvent, int childID);

[C++] protected: void AccessibilityNotifyClients(AccessibleEvents accEvent, int childID);

[VB] Protected Sub AccessibilityNotifyClients(ByVal accEvent As AccessibleEvents, ByVal childID As Integer)

[JScript] protected function AccessibilityNotifyClients(accEvent : AccessibleEvents, childID : int);

## Description

Notifies the accessibility client applications of the specified **System.Windows.Forms.AccessibleEvents** for the specified child control.

You must call the **System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method for each **System.Windows.Forms.AccessibleEvents** object the accessibility client applications are to be notified of. The **System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method is typically called when a property is set or from within an event handler. For example, you might call the **System.Windows.Forms.Control.ControlAccessibleObject.NotifyEvents**

method and pass in **System.Windows.Forms.AccessibleEvents.Hide** from within the event handler for the **System.Windows.Forms.Control.VisibleChanged** event. The **System.Windows.Forms.AccessibleEvents** object to notify the accessibility client applications of. The child **System.Windows.Forms.Control** to notify of the accessible event.

*oooooooo)BeginInvoke*

[C#]        public        IAsyncResult        BeginInvoke(Delegate        method);

[C++]       public:        IAsyncResult\*        BeginInvoke(Delegate\*        method);

[VB] Public Function BeginInvoke(ByVal method As Delegate) As IAsyncResult

[JScript] public function BeginInvoke(method : Delegate) : IAsyncResult;

Executes a delegate asynchronously on the thread that the control's underlying handle was created on

### *Description*

Executes the specified delegate asynchronously on the thread that the control's underlying handle was created on

*Return Value:* An **System.IAsyncResult** object that represents the result of the **System.Windows.Forms.Control.BeginInvoke(System.Delegate)** operation.

The delegate is called asynchronously and this method returns immediately. You may call this from any thread, even the thread that owns the control's handle. If the control's handle does not exist yet, this will follow up the control's parent chain until it finds a control or form that does have a window handle. If no appropriate handle can be found,

**System.Windows.Forms.Control.BeginInvoke(System.Delegate)** will throw an exception. Exceptions within the delegate method are considered untrapped and will be sent to the application's untrapped exception handler. A delegate to a method that takes no parameters.



## ppppppppp)BeginInvoke

```
[C#] public IAsyncResult BeginInvoke(Delegate method, object[] args);
[C++] public: __sealed IAsyncResult* BeginInvoke(Delegate* method, Object*
args
__gc[]);
[VB] NotOverridable Public Function BeginInvoke(ByVal method As Delegate,
ByVal args() As Object) As IAsyncResult
[JScript] public function BeginInvoke(method : Delegate, args : Object[]) :
IAsyncResult;
```

### Description

Executes the specified delegate asynchronously with the specified arguments, on the thread that the control's underlying handle was created on.

**Return Value:** An **System.IAsyncResult** object that represents the result of the **System.Windows.Forms.Control.BeginInvoke(System.Delegate)** operation.

The delegate is called asynchronously and this method returns immediately. You may call this from any thread, even the thread that owns the control's handle. If the control's handle does not exist yet, this will follow up the control's parent chain until it finds a control or form that does have a window handle. If no appropriate handle can be found,

**System.Windows.Forms.Control.BeginInvoke(System.Delegate)** will throw an exception. Exceptions within the delegate method are considered untrapped and will be sent to the application's untrapped exception handler. A delegate to a method that takes parameters of the same number and type that are contained in the *args* parameter. An array of objects to pass as arguments to the given method. This can be null if no arguments are needed.

## qqqqqqqqq)BringToFront

```
[C#] public void BringToFront();
[C++] public: void BringToFront();
```

[VB]	Public	Sub	BringToFront()
[JScript]	public	function	BringToFront();

#### *Description*

Brings the control to the front of the z-order.

The control is moved to the top of the z-order. If the control is a child of another control, the child control is moved to the top of the z-order.

#### *rrrrrrrr)Contains*

[C#]	public	bool	Contains(Control	ctl);
[C++]	public:	bool	Contains(Control*	ctl);
[VB]	Public	Function	Contains(ByVal ctl As Control)	As Boolean
[JScript]	public	function	Contains(ctl : Control)	: Boolean;

#### *Description*

Retrieves a value indicating whether the specified control is a child of the control.

**Return Value:** **true** if the specified control is a child of the control; otherwise, **false** . The **System.Windows.Forms.Control** to evaluate.

#### *ssssssss)CreateAccessibilityInstance*

[C#]	protected	virtual	AccessibleObject	CreateAccessibilityInstance();
[C++]	protected:	virtual	AccessibleObject*	CreateAccessibilityInstance();
[VB]	Overridable	Protected	Function	CreateAccessibilityInstance() As
			AccessibleObject	
[JScript]	protected	function	CreateAccessibilityInstance()	: AccessibleObject;

## Description

Creates a new instance of the accessibility object for the control.

*Return Value:* A new instance of the

**System.Windows.Forms.AccessibleObject** assigned to the control.

If you do not explicitly call the

**System.Windows.Forms.Control.CreateAccessibilityInstance** method it will be called when the

**System.Windows.Forms.Control.AccessibilityObject** property is referenced.

## ttttttt) CreateControl

[C#]	public	void	CreateControl();
[C++]	public:	void	CreateControl();
[VB]	Public	Sub	CreateControl()
[JScript]	public	function	CreateControl();

## Description

Forces the creation of the control including the creation of the handle and any child controls.

## uuuuuuuuu)CreateControlsInstance

[C#]	protected	virtual	ControlCollection	CreateControlsInstance();
[C++]	protected:	virtual	ControlCollection*	CreateControlsInstance();
[VB]	Overridable	Protected	Function	CreateControlsInstance() As ControlCollection
[JScript]	protected	function	CreateControlsInstance()	: ControlCollection;

## Description

Creates a new instance of the control collection for the control.

*Return Value:* A new instance of

**System.Windows.Forms.Control.ControlCollection** assigned to the control.

The base class version of this method should not be called by a derived class.

~~~~~)CreateGraphics

[C#]                    public                    Graphics                    CreateGraphics();

[C++]                    public:                    Graphics\*                    CreateGraphics();

[VB]           Public           Function           CreateGraphics()           As           Graphics

[JScript]   public   function   CreateGraphics()   :   Graphics;   Creates   the

**System.Drawing.Graphics**           object           for           the           control.

## Description

Creates the **System.Drawing.Graphics** object for the control.

*Return Value:* The **System.Drawing.Graphics** object for the control.

The control's brush, font, foreground color, and background color become the default values for the **System.Drawing.Graphics** object. The returned **System.Drawing.Graphics** object must be disposed through a call to its **System.Drawing.Graphics.Dispose** method when it is no longer needed. The **System.Drawing.Graphics** object is only valid for the duration of the current window's message.

~~~~~)CreateHandle

[C#]                    protected                    virtual                    void                    CreateHandle();

[C++]                    protected:                    virtual                    void                    CreateHandle();

[VB]                    Overridable                    Protected                    Sub                    CreateHandle()

[JScript]                      protected                      function                      CreateHandle();

*Description*

Creates a handle for this control.

When overriding **System.Windows.Forms.Control.CreateHandle** in a derived class, be sure to call the base class's **System.Windows.Forms.Control.CreateHandle** method.

*xxxxxxxxx)DefWndProc*

[C#]            protected            virtual            void            DefWndProc(ref            Message            m);

[C++]            protected:            virtual            void            DefWndProc(Message\*            m);

[VB]            Overridable            Protected            Sub            DefWndProc(ByRef            m            As            Message)

[JScript]            protected            function            DefWndProc(m            :            Message);

*Description*

Sends the specified message to the default window procedure.

For more information about processing Windows messages, see the **WindowProc** function documentation in the Windows Platform SDK reference located in the MSDN Library. The **System.Windows.Forms.Message** to process.

*yyyyyyyyy)DestroyHandle*

[C#]            protected            virtual            void            DestroyHandle();

[C++]            protected:            virtual            void            DestroyHandle();

[VB]            Overridable            Protected            Sub            DestroyHandle()

[JScript]            protected            function            DestroyHandle();

## Description

Destroys the handle associated with this control.

When overriding **System.Windows.Forms.Control.DestroyHandle** in a derived class, be sure to call the base class's **System.Windows.Forms.Control.DestroyHandle** method.

## zzzzzzzzzz)Dispose

[C#]       protected       override       void       Dispose(bool       disposing);

[C++]       protected:       void       Dispose(bool       disposing);

[VB]   Overrides   Protected   Sub   Dispose(ByVal   disposing   As   Boolean)

[JScript] protected override function Dispose(disposing : Boolean); Releases all resources used by the **System.Windows.Forms.Control**.

## Description

Releases the unmanaged resources used by the **System.Windows.Forms.Control** and optionally releases the managed resources.

This method is called by the public method and the **System.Object.Finalize** method. **true** to release both managed and unmanaged resources; **false** to release only unmanaged resources.

## aaaaaaaaaa)DoDragDrop

[C#]   public   DragDropEffects   DoDragDrop(object   data,   DragDropEffects  
allowedEffects);

[C++]   public:   DragDropEffects   DoDragDrop(Object\*   data,   DragDropEffects  
allowedEffects);

```

1 [VB] Public Function DoDragDrop(ByVal data As Object, ByVal allowedEffects
2 As DragDropEffects) As DragDropEffects
3 [JScript] public function DoDragDrop(data : Object, allowedEffects :
4 DragDropEffects) : DragDropEffects;

```

### Description

Begins a drag and drop operation.

*Return Value:* A value from the **System.Windows.Forms.DragDropEffects** enumeration that represents the final effect that was performed during the drag and drop operation.

The *allowedEffects* parameter determines which drag operations can occur. If the drag operation needs to interoperate with applications in another process, data should either be a base managed class ( **System.String** , **System.Drawing.Bitmap** , or **System.Drawing.Imaging.Metafile** ), or an object that implements **System.Runtime.Serialization.ISerializable** or **System.Windows.Forms.IDataObject** . The data to drag. One of the **System.Windows.Forms.DragDropEffects** values.

### bbbbbbbbbb)EndInvoke

```

16 [C#] public object EndInvoke(IAsyncResult asyncResult);
17 [C++] public: __sealed Object* EndInvoke(IAsyncResult* asyncResult);
18 [VB] NotOverridable Public Function EndInvoke(ByVal asyncResult As
19 IAsyncResult) As Object
20 [JScript] public function EndInvoke(asyncResult : IAsyncResult) : Object;

```

### Description

Retrieves the return value of the asynchronous operation represented by the IAsyncResult interface passed.

*Return Value:* The **System.Object** generated by the asynchronous operation.

If the async operation has not been completed, this function will block until the result is available. The **System.IAsyncResult** interface that represents a specific invoke asynchronous operation, returned when calling **System.Windows.Forms.Control.BeginInvoke(System.Delegate)** .

#### *ccccccccc)FindForm*

|           |         |          |                    |
|-----------|---------|----------|--------------------|
| [C#]      | public  | Form     | FindForm();        |
| [C++]     | public: | Form*    | FindForm();        |
| [VB]      | Public  | Function | FindForm() As Form |
| [JScript] | public  | function | FindForm() : Form; |

#### *Description*

Retrieves the form that the control is on.

*Return Value:* The **System.Windows.Forms.Form** that the control is on.

The control's **System.Windows.Forms.Control.Parent** property value may not be the same as the **System.Windows.Forms.Form** returned by **System.Windows.Forms.Control.FindForm** method.

#### *ddddddddddd)Focus*

|           |         |          |                    |
|-----------|---------|----------|--------------------|
| [C#]      | public  | bool     | Focus();           |
| [C++]     | public: | bool     | Focus();           |
| [VB]      | Public  | Function | Focus() As Boolean |
| [JScript] | public  | function | Focus() : Boolean; |

#### *Description*

Sets input focus to the control.

*Return Value:* **true** if the input focus request was successful; otherwise, **false** .

The **System.Windows.Forms.Control.Focus** method returns **true** if the control successfully received input focus. The control can have the input focus



while not displaying any visual cues of having the focus. This behavior is primarily observed by the nonselectable controls listed below, or any controls derived from them.

### *eeeeeeeeee)FromChildHandle*

[C#] public static Control FromChildHandle(IntPtr handle);

[C++] public: static Control\* FromChildHandle(IntPtr handle);

[VB] Public Shared Function FromChildHandle(ByVal handle As IntPtr) As Control

[JScript] public static function FromChildHandle(handle : IntPtr) : Control;

### *Description*

Retrieves the control that contains the specified handle.

**Return Value:** The **System.Windows.Forms.Control** that represents the control associated with the specified handle; returns **null** if no control with the specified handle is found.

This method searches up the window handle parent chain until it finds a handle that is associated with a control. This method is more robust than the **System.Windows.Forms.Control.FromHandle(System.IntPtr)** method, because it correctly returns controls that own more than one handle. The window handle (HWND) to search for.

### *fffffffff)FromHandle*

[C#] public static Control FromHandle(IntPtr handle);

[C++] public: static Control\* FromHandle(IntPtr handle);

[VB] Public Shared Function FromHandle(ByVal handle As IntPtr) As Control

[JScript] public static function FromHandle(handle : IntPtr) : Control;

### *Description*

Returns the control that is currently associated with the specified handle.

*Return Value:* A **System.Windows.Forms.Control** that represents the control associated with the specified handle; returns **null** if no control with the specified handle is found.

Use the

**System.Windows.Forms.Control.FromChildHandle(System.IntPtr)**

method if you need to correctly return controls that own more than one handle.

The window handle (HWND) to search for.

*gggggggggg)GetChildAtPoint*

[C#]            public            Control            GetChildAtPoint(Point            pt);

[C++]            public:            Control\*            GetChildAtPoint(Point            pt);

[VB]   Public   Function   GetChildAtPoint(ByVal pt As Point) As Control

[JScript]   public   function   GetChildAtPoint(pt : Point) : Control;

### *Description*

Retrieves the child control that is located at the specified coordinates.

*Return Value:* A **System.Windows.Forms.Control** that represents the control that is located at the specified point.

If there is no control at the specified point other than this control, the **System.Windows.Forms.Control.GetChildAtPoint(System.Drawing.Point)** method returns **null**. A **System.Drawing.Point** that contains the coordinates where you want to look for a control. Coordinates are expressed relative to the upper-left corner of the control's client area.

*hhhhhhhhhh)GetContainerControl*

[C#]            public            IContainerControl            GetContainerControl();

[C++]            public:            IContainerControl\*            GetContainerControl();

[VB]   Public   Function   GetContainerControl() As IContainerControl

[JScript]   public   function   GetContainerControl() : IContainerControl;

## Description

Returns the next **System.Windows.Forms.ContainerControl** up the control's chain of parent controls.

*Return Value:* An object implementing the **System.Windows.Forms.IContainerControl** interface, that represents the parent of the **System.Windows.Forms.Control** .

### *iiiiiiii)GetNextControl*

```
[C#] public Control GetNextControl(Control ctl, bool forward);
```

```
[C++] public: Control* GetNextControl(Control* ctl, bool forward);
```

```
[VB] Public Function GetNextControl(ByVal ctl As Control, ByVal forward As Boolean) As Control
```

```
[JScript] public function GetNextControl(ctl : Control, forward : Boolean) : Control;
```

## Description

Retrieves the next control forward, or back, in the tab order of child controls.

*Return Value:* The next **System.Windows.Forms.Control** in the tab order. The **System.Windows.Forms.Control** to start the search with. **true** to search forward in the tab order; otherwise, **false**.

### *jjjjjjjj)GetStyle*

```
[C#] protected bool GetStyle(ControlStyles flag);
```

```
[C++] protected: bool GetStyle(ControlStyles flag);
```

```
[VB] Protected Function GetStyle(ByVal flag As ControlStyles) As Boolean
```

```
[JScript] protected function GetStyle(flag : ControlStyles) : Boolean;
```

## Description

Retrieves the value of the specified control style bit for the control.

**Return Value:** **true** if specified control sytle bit is set to **true** ; otherwise, **false** .

Control style bit flags are used to categorize supported behavior. A control can enable a style by calling the **System.Windows.Forms.Control.SetStyle(System.Windows.Forms.ControlStyles,System.Boolean)** method and passing in the appropriate **System.Windows.Forms.ControlStyles** bit and the boolean value set the bit to. To determine the value assigned to a specified **System.Windows.Forms.ControlStyles** bit, use the **System.Windows.Forms.Control.GetStyle(System.Windows.Forms.ControlStyles)** method and pass in the **System.Windows.Forms.ControlStyles** member to evaluate. The **System.Windows.Forms.ControlStyles** bit to return the value from.

*kkkkkkkkkk)GetTopLevel*

|           |            |          |                          |
|-----------|------------|----------|--------------------------|
| [C#]      | protected  | bool     | GetTopLevel();           |
| [C++]     | protected: | bool     | GetTopLevel();           |
| [VB]      | Protected  | Function | GetTopLevel() As Boolean |
| [JScript] | protected  | function | GetTopLevel() : Boolean; |

## Description

Determines if the control is a top-level control.

**Return Value:** **true** if the **System.Windows.Forms.Control** is a top-level control; otherwise, **false** .

*lllllllll)Hide*

|       |         |      |         |
|-------|---------|------|---------|
| [C#]  | public  | void | Hide(); |
| [C++] | public: | void | Hide(); |

|           |        |          |         |
|-----------|--------|----------|---------|
| [VB]      | Public | Sub      | Hide()  |
| [JScript] | public | function | Hide(); |

#### Description

Conceals the control from the user.

Hiding the control is equal to setting the **System.Windows.Forms.Control.Visible** property to **false** . After the **System.Windows.Forms.Control.Hide** method is called the **System.Windows.Forms.Control.Visible** property returns a value of **false** until the **System.Windows.Forms.Control.Show** method is called.

#### *mmmmmmmmmm)InitLayout*

|           |             |           |      |               |
|-----------|-------------|-----------|------|---------------|
| [C#]      | protected   | virtual   | void | InitLayout(); |
| [C++]     | protected:  | virtual   | void | InitLayout(); |
| [VB]      | Overridable | Protected | Sub  | InitLayout()  |
| [JScript] | protected   | function  |      | InitLayout(); |

#### Description

Called after the control has been added to another container.

#### *nnnnnnnnnn)Invalidate*

|           |         |          |               |
|-----------|---------|----------|---------------|
| [C#]      | public  | void     | Invalidate(); |
| [C++]     | public: | void     | Invalidate(); |
| [VB]      | Public  | Sub      | Invalidate()  |
| [JScript] | public  | function | Invalidate(); |

## Description

Invalidates a specific region of the control and causes a paint message to be sent to the control.

### *ooooooooo)Invalidate*

[C#]        public        void        Invalidate(bool        invalidateChildren);

[C++]        public:        void        Invalidate(bool        invalidateChildren);

[VB]    Public    Sub    Invalidate(ByVal    invalidateChildren    As    Boolean)

[JScript]    public    function    Invalidate(invalidateChildren    :    Boolean);

## Description

Invalidates a specific region of the control and causes a paint message to be sent to the control. Optionally, invalidates the child controls assigned to the control. **true** to invalidate the control's child controls; otherwise, **false**.

### *pppppppppp)Invalidate*

[C#]        public        void        Invalidate(Rectangle        rc);

[C++]        public:        void        Invalidate(Rectangle        rc);

[VB]    Public    Sub    Invalidate(ByVal    rc    As    Rectangle)

[JScript]    public    function    Invalidate(rc    :    Rectangle);

## Description

Invalidates the specified region of the control (adds it to the control's update region, which is the area that will be repainted at the next paint operation), and causes a paint message to be sent to the control. A

**System.Drawing.Rectangle** object that represents the region to invalidate.

## *qqqqqqqqq)Invalidate*

[C#]            public            void            Invalidate(Region            region);

[C++]           public:            void            Invalidate(Region\*            region);

[VB]    Public    Sub    Invalidate(ByVal    region    As    Region)

[JScript] public function Invalidate(region : Region); Invalidates a specific region of the control and causes a paint message to be sent to the control.

### *Description*

Invalidates the specified region of the control (adds it to the control's update region, which is the area that will be repainted at the next paint operation), and causes a paint message to be sent to the control.

Calling this method does not force a synchronous paint; to force a synchronous paint, call the **System.Windows.Forms.Control.Update** method after calling the

**System.Windows.Forms.Control.Invalidate(System.Drawing.Region)** method. If this method is called with no parameters, the entire client area is added to the update region. The **System.Drawing.Region** to invalidate.

## *rrrrrrrrrr)Invalidate*

[C#]    public    void    Invalidate(Rectangle rc,    bool    invalidateChildren);

[C++]    public:    void    Invalidate(Rectangle rc,    bool    invalidateChildren);

[VB] Public Sub Invalidate(ByVal rc As Rectangle, ByVal invalidateChildren As Boolean)

[JScript] public function Invalidate(rc : Rectangle, invalidateChildren : Boolean);

### *Description*

Invalidates the specified region of the control (adds it to the control's update region, which is the area that will be repainted at the next paint operation), and

causes a paint message to be sent to the control. Optionally, invalidates the child controls assigned to the control. A **System.Drawing.Rectangle** object that represents the region to invalidate. **true** to invalidate the control's child controls; otherwise, **false**.

### *sssssssss)Invalidate*

[C#] public void Invalidate(Region region, bool invalidateChildren);

[C++] public: void Invalidate(Region\* region, bool invalidateChildren);

[VB] Public Sub Invalidate(ByVal region As Region, ByVal invalidateChildren As Boolean)

[JScript] public function Invalidate(region : Region, invalidateChildren : Boolean);

### *Description*

Invalidates the specified region of the control (adds it to the control's update region, which is the area that will be repainted at the next paint operation), and causes a paint message to be sent to the control. Optionally, invalidates the child controls assigned to the control. The **System.Drawing.Region** to invalidate. **true** to invalidate the control's child controls; otherwise, **false**.

### *ttttttttt)Invoke*

[C#] public object Invoke(Delegate method);

[C++] public: Object\* Invoke(Delegate\* method);

[VB] Public Function Invoke(ByVal method As Delegate) As Object

[JScript] public function Invoke(method : Delegate) : Object; Executes a delegate on the thread that owns this control's underlying window handle.

### *Description*



Executes the specified delegate on the thread that owns this control's underlying window handle.

*Return Value:* The return value from the delegate being invoked, or **null** if the delegate has no return value.

It is an error to call this on the same thread that the control belongs to. If the control's handle does not exist yet, this will follow up the control's parent chain until it finds a control or form that does have a window handle. If no appropriate handle can be found, the

**System.Windows.Forms.Control.Invoke(System.Delegate)** method will throw an exception. Exceptions that are raised during the call will be propagated back to the caller. A delegate that contains a method to be called in the control's thread context.

*uuuuuuuuuu)Invoke*

[C#] public object Invoke(Delagate method, object[] args);

[C++] public: \_\_sealed Object\* Invoke(Delagate\* method, Object\* args \_\_gc[]);

[VB] NotOverridable Public Function Invoke(ByVal method As Delagate, ByVal args() As Object) As Object

[JScript] public function Invoke(method : Delagate, args : Object[]) : Object;

### Description

Executes the specified delegate, on the thread that owns this control's underlying window handle, with the specified list of arguments.

*Return Value:* The return value from the delegate being invoked, or **null** if the delegate has no return value.

It is an error to call this on the same thread that the control belongs to. If the control's handle does not exist yet, this will follow up the control's parent chain until it finds a control or form that does have a window handle. If no appropriate handle can be found, the

**System.Windows.Forms.Control.Invoke(System.Delegate)** method will throw an exception. Exceptions that are raised during the call will be propagated back to the caller. A delegate to a method that takes parameters of the same number and type that are contained in args. An array of objects to pass as arguments to the given method. This can be null if no arguments are needed.

## vvvvvvvvvv)InvokeGotFocus

```
[C#] protected void InvokeGotFocus(Control toInvoke, EventArgs e);
[C++] protected: void InvokeGotFocus(Control* toInvoke, EventArgs* e);
[VB] Protected Sub InvokeGotFocus(ByVal toInvoke As Control, ByVal e As
EventArgs)
[JScript] protected function InvokeGotFocus(toInvoke : Control, e : EventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.GotFocus** event. The **System.Windows.Forms.Control** to assign the event to. An **System.EventArgs** that contains the event data.

## wwwwwwwwww)InvokeLostFocus

```
[C#] protected void InvokeLostFocus(Control toInvoke, EventArgs e);
[C++] protected: void InvokeLostFocus(Control* toInvoke, EventArgs* e);
[VB] Protected Sub InvokeLostFocus(ByVal toInvoke As Control, ByVal e As
EventArgs)
[JScript] protected function InvokeLostFocus(toInvoke : Control, e : EventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.LostFocus** event. The **System.Windows.Forms.Control** to assign the event to. An **System.EventArgs** that contains the event data.

## xxxxxxxxxx)InvokeOnClick

```
[C#] protected void InvokeOnClick(Control toInvoke, EventArgs e);
```

```

1 [C++] protected: void InvokeOnClick(Control* toInvoke, EventArgs* e);
2 [VB] Protected Sub InvokeOnClick(ByVal toInvoke As Control, ByVal e As
3 EventArgs)
4 [JScript] protected function InvokeOnClick(toInvoke : Control, e : EventArgs);
5

```

### *Description*

Raises the **System.Windows.Forms.Control.Click** event for a specific control. The **System.Windows.Forms.Control** to assign the **System.Windows.Forms.Control.Click** event to. An **System.EventArgs** that contains the event data.

### *yyyyyyyyyy)InvokePaint*

```

11 [C#] protected void InvokePaint(Control c, PaintEventArgs e);
12 [C++] protected: void InvokePaint(Control* c, PaintEventArgs* e);
13 [VB] Protected Sub InvokePaint(ByVal c As Control, ByVal e As
14 PaintEventArgs)
15 [JScript] protected function InvokePaint(c : Control, e : PaintEventArgs);
16

```

### *Description*

Raises the **System.Windows.Forms.Control.Paint** event for a specific control. The **System.Windows.Forms.Control** to assign the **System.Windows.Forms.Control.Paint** event to. An **System.Windows.Forms.PaintEventArgs** that contains the event data.

### *zzzzzzzzzz)InvokePaintBackground*

```

23 [C#] protected void InvokePaintBackground(Control c, PaintEventArgs e);
24 [C++] protected: void InvokePaintBackground(Control* c, PaintEventArgs* e);
25

```

[VB] Protected Sub InvokePaintBackground(ByVal c As Control, ByVal e As PaintEventArgs)

[JScript] protected function InvokePaintBackground(c : Control, e : PaintEventArgs);

### *Description*

Raises the **PaintBackground** event for a specific control. The **System.Windows.Forms.Control** to assign the **System.Windows.Forms.Control.Paint** event to. An **System.Windows.Forms.PaintEventArgs** that contains the event data.

### *aaaaaaaaaaaa)IsInputChar*

[C#] protected virtual bool IsInputChar(char charCode);

[C++] protected: virtual bool IsInputChar(\_\_wchar\_t charCode);

[VB] Overridable Protected Function IsInputChar(ByVal charCode As Char) As Boolean

[JScript] protected function IsInputChar(charCode : Char) : Boolean;

### *Description*

Determines if a character is an input character that the control recognizes.

*Return Value:* **true** if the character should be sent directly to control and not preprocessed; otherwise, **false** .

This method is called during window message preprocessing to determine whether the given input character should be preprocessed or sent directly to the control. If the

**System.Windows.Forms.Control.IsInputChar(System.Char)** method returns **true** , the specified character is sent directly to the control. However, if the method returns **false** , the character is preprocessed and only sent to the control if it is not consumed by the preprocessing phase. The preprocessing of a character includes checking whether the character is a mnemonic of another control. The character to test.

## *bbbbbbbbbbb)IsInputKey*

```
[C#]    protected    virtual    bool    IsInputKey(Keys    keyData);
[C++]    protected:    virtual    bool    IsInputKey(Keys    keyData);
[VB]    Overridable Protected Function IsInputKey(ByVal keyData As Keys) As
Boolean
[JScript]    protected    function    IsInputKey(keyData : Keys) : Boolean;
```

### *Description*

Determines whether the specified key is a regular input key or a special key that requires preprocessing.

**Return Value:** **true** if the specified key is a regular input key; otherwise, **false** .

Call this method during window-message preprocessing to determine whether the specified key is a regular input key that should be sent directly to the control or a special key (such as PAGE UP and PAGE DOWN) that should be preprocessed. In the latter case, send the key to the control only if it is not consumed by the preprocessing phase. One of the **System.Windows.Forms.Keys** values.

## *ccccccccccc)IsMnemonic*

```
[C#]    public    static    bool    IsMnemonic(char    charCode,    string    text);
[C++]    public:    static    bool    IsMnemonic(__wchar_t    charCode,    String*    text);
[VB]    Public Shared Function IsMnemonic(ByVal charCode As Char, ByVal text
As
String) As
Boolean
[JScript]    public static function IsMnemonic(charCode : Char, text : String) :
Boolean;
```

### *Description*

Determines if the specified character is the mnemonic character assigned to the control in the specified string.

*Return Value:* **true** if the *charCode* character is the mnemonic character assigned to the control; otherwise, **false** .

The mnemonic character is the character immediately following the first instance of "&" in a **System.String** . The character to test. The **System.String** to search.

*ddddddddddd)NotifyInvalidate*

[C#] protected virtual void NotifyInvalidate(Rectangle invalidatedArea);

[C++] protected: virtual void NotifyInvalidate(Rectangle invalidatedArea);

[VB] Overridable Protected Sub NotifyInvalidate(ByVal invalidatedArea As Rectangle)

[JScript] protected function NotifyInvalidate(invalidatedArea : Rectangle);

#### *Description*

Raises the **System.Windows.Forms.Control.Invalidated** event with a specified region of the control to invalidate. A **System.Drawing.Rectangle** representing the area to invalidate.

*eeeeeeeeeee)OnBackColorChanged*

[C#] protected virtual void OnBackColorChanged(EventArgs e);

[C++] protected: virtual void OnBackColorChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnBackColorChanged(ByVal e As EventArgs)

[JScript] protected function OnBackColorChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.BackColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *OnBackgroundImageChanged*

[C#] protected virtual void OnBackgroundImageChanged(EventArgs e);

[C++] protected: virtual void OnBackgroundImageChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnBackgroundImageChanged(ByVal e As EventArgs)

[JScript] protected function OnBackgroundImageChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.BackgroundImageChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *OnBindingContextChanged*

[C#] protected virtual void OnBindingContextChanged(EventArgs e);

[C++] protected: virtual void OnBindingContextChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnBindingContextChanged(ByVal e As EventArgs)

[JScript] protected function OnBindingContextChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.BindingContextChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *hhhhhhhhh)OnCausesValidationChanged*

[C#] protected virtual void OnCausesValidationChanged(EventArgs e);

[C++] protected: virtual void OnCausesValidationChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnCausesValidationChanged(ByVal e As EventArgs)

[JScript] protected function OnCausesValidationChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.CausesValidationChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *iiiiiiiiii)OnChangeUICues*

[C#] protected virtual void OnChangeUICues(UICuesEventArgs e);

[C++] protected: virtual void OnChangeUICues(UICuesEventArgs\* e);

[VB] Overridable Protected Sub OnChangeUICues(ByVal e As UICuesEventArgs)

[JScript] protected function OnChangeUICues(e : UICuesEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.ChangeUICues** event.



Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.UICuesEventArgs** that contains the event data.

### *jjjjjjjjj)OnClick*

```
[C#]      protected      virtual      void      OnClick(EventArgs      e);
[C++]      protected:      virtual      void      OnClick(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnClick(ByVal e As EventArgs)
[JScript]      protected      function      OnClick(e      :      EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Control.Click** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *kkkkkkkkkk)OnContextMenuChanged*

```
[C#]      protected      virtual      void      OnContextMenuChanged(EventArgs      e);
[C++]      protected:      virtual      void      OnContextMenuChanged(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnContextMenuChanged(ByVal e As
EventArgs)
[JScript]      protected      function      OnContextMenuChanged(e      :      EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Control.ContextMenuChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *lllllllll)OnControlAdded*

```
[C#]    protected    virtual    void    OnControlAdded(ControlEventArgs e);
[C++]    protected:    virtual    void    OnControlAdded(ControlEventArgs* e);
[VB]    Overridable Protected Sub OnControlAdded(ByVal e As ControlEventArgs)
[JScript]    protected    function    OnControlAdded(e : ControlEventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.ControlAdded** event.

Called when a child control is added to this control. A **System.Windows.Forms.ControlEventArgs** that contains the event data.

### *mmmmmmmmmm)OnControlRemoved*

```
[C#]    protected    virtual    void    OnControlRemoved(ControlEventArgs e);
[C++]    protected:    virtual    void    OnControlRemoved(ControlEventArgs* e);
[VB]    Overridable    Protected    Sub    OnControlRemoved(ByVal e As
ControlEventArgs)
[JScript]    protected    function    OnControlRemoved(e : ControlEventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.ControlRemoved** event.

Called when a child control is removed from this control. A **System.Windows.Forms.ControlEventArgs** that contains the event data.

### *nnnnnnnnnn)OnCreateControl*

```
[C#]    protected    virtual    void    OnCreateControl();
```

|           |             |           |      |                    |
|-----------|-------------|-----------|------|--------------------|
| [C++]     | protected:  | virtual   | void | OnCreateControl(); |
| [VB]      | Overridable | Protected | Sub  | OnCreateControl()  |
| [JScript] | protected   | function  |      | OnCreateControl(); |

#### Description

Raises the **System.Windows.Forms.Control.CreateControl** event.

Called when the control is first created.

*oooooooooooo)OnCursorChanged*

|           |             |           |      |                                       |
|-----------|-------------|-----------|------|---------------------------------------|
| [C#]      | protected   | virtual   | void | OnCursorChanged(EventArgs e);         |
| [C++]     | protected:  | virtual   | void | OnCursorChanged(EventArgs* e);        |
| [VB]      | Overridable | Protected | Sub  | OnCursorChanged(ByVal e As EventArgs) |
| [JScript] | protected   | function  |      | OnCursorChanged(e : EventArgs);       |

#### Description

Raises the **System.Windows.Forms.Control.CursorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*ppppppppppp)OnDockChanged*

|           |             |           |      |                                     |
|-----------|-------------|-----------|------|-------------------------------------|
| [C#]      | protected   | virtual   | void | OnDockChanged(EventArgs e);         |
| [C++]     | protected:  | virtual   | void | OnDockChanged(EventArgs* e);        |
| [VB]      | Overridable | Protected | Sub  | OnDockChanged(ByVal e As EventArgs) |
| [JScript] | protected   | function  |      | OnDockChanged(e : EventArgs);       |

## Description

Raises the **System.Windows.Forms.Control.DockChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*qqqqqqqqqq)OnDoubleClick*

[C#]      protected      virtual      void      OnDoubleClick(EventArgs      e);

[C++]      protected:      virtual      void      OnDoubleClick(EventArgs\*      e);

[VB]      Overridable      Protected      Sub      OnDoubleClick(ByVal e As EventArgs)

[JScript]      protected      function      OnDoubleClick(e      :      EventArgs);

## Description

Raises the **System.Windows.Forms.Control.DoubleClick** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*rrrrrrrrrr)OnDragDrop*

[C#]      protected      virtual      void      OnDragDrop(DragEventArgs      drgevent);

[C++]      protected:      virtual      void      OnDragDrop(DragEventArgs\*      drgevent);

[VB]      Overridable      Protected      Sub      OnDragDrop(ByVal drgevent As  
DragEventArgs)

[JScript]      protected      function      OnDragDrop(drgevent      :      DragEventArgs);

## Description

Raises the **System.Windows.Forms.Control.DragDrop** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DragEventArgs** that contains the event data.

### *ssssssssss)OnDragEnter*

[C#] protected virtual void OnDragEnter(DragEventArgs drgevent);

[C++] protected: virtual void OnDragEnter(DragEventArgs\* drgevent);

[VB] Overridable Protected Sub OnDragEnter(ByVal drgevent As DragEventArgs)

[JScript] protected function OnDragEnter(drgevent : DragEventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.DragEnter** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DragEventArgs** that contains the event data.

### *ttttttttt)OnDragLeave*

[C#] protected virtual void OnDragLeave(EventArgs e);

[C++] protected: virtual void OnDragLeave(EventArgs\* e);

[VB] Overridable Protected Sub OnDragLeave(ByVal e As EventArgs)

[JScript] protected function OnDragLeave(e : EventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.DragLeave** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## uuuuuuuuuuuu)OnDragOver

```
[C#]    protected    virtual    void    OnDragOver(DragEventArgs    drgevent);
[C++]    protected:    virtual    void    OnDragOver(DragEventArgs*    drgevent);
[VB]    Overridable    Protected    Sub    OnDragOver(ByVal    drgevent    As
DragEventArgs)
[JScript]    protected    function    OnDragOver(drgevent    :    DragEventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.DragOver** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DragEventArgs** that contains the event data.

## vvvvvvvvvvvv)OnEnabledChanged

```
[C#]    protected    virtual    void    OnEnabledChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnEnabledChanged(EventArgs*    e);
[VB]    Overridable    Protected    Sub    OnEnabledChanged(ByVal    e    As    EventArgs)
[JScript]    protected    function    OnEnabledChanged(e    :    EventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.EnabledChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## www)OnEnter

```
[C#]      protected      virtual      void      OnEnter(EventArgs      e);
[C++]      protected:      virtual      void      OnEnter(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnEnter(ByVal e As EventArgs)
[JScript]      protected      function      OnEnter(e      :      EventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.Enter** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## xxxxxxxxxx)OnFontChanged

```
[C#]      protected      virtual      void      OnFontChanged(EventArgs      e);
[C++]      protected:      virtual      void      OnFontChanged(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnFontChanged(ByVal e As EventArgs)
[JScript]      protected      function      OnFontChanged(e      :      EventArgs);
```

### Description

Raises the **System.Windows.Forms.Control.FontChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## yyyyyyyyyyy)OnForeColorChanged

```
[C#]      protected      virtual      void      OnForeColorChanged(EventArgs      e);
[C++]      protected:      virtual      void      OnForeColorChanged(EventArgs*      e);
```

[VB] Overridable Protected Sub OnForeColorChanged(ByVal e As EventArgs)

[JScript] protected function OnForeColorChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.ForeColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

zzzzzzzzzz)OnGiveFeedback

[C#] protected virtual void OnGiveFeedback(GiveFeedbackEventArgs gfbevent);

[C++] protected: virtual void OnGiveFeedback(GiveFeedbackEventArgs\* gfbevent);

[VB] Overridable Protected Sub OnGiveFeedback(ByVal gfbevent As GiveFeedbackEventArgs)

[JScript] protected function OnGiveFeedback(gfbevent : GiveFeedbackEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.GiveFeedback** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.GiveFeedbackEventArgs** that contains the event data.

aaaaaaaaaaaa)OnGotFocus

[C#] protected virtual void OnGotFocus(EventArgs e);

[C++] protected: virtual void OnGotFocus(EventArgs\* e);



1 [VB] Overridable Protected Sub OnGotFocus(ByVal e As EventArgs)

2 [JScript] protected function OnGotFocus(e : EventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.Control.GotFocus** event.

6 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

7 *bbbbbbbbbbbb)OnHandleCreated*

8  
9 [C#] protected virtual void OnHandleCreated(EventArgs e);

10 [C++] protected: virtual void OnHandleCreated(EventArgs\* e);

11 [VB] Overridable Protected Sub OnHandleCreated(ByVal e As EventArgs)

12 [JScript] protected function OnHandleCreated(e : EventArgs);

13  
14 *Description*

15 Raises the **System.Windows.Forms.Control.HandleCreated** event.

16 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

17  
18 *ccccccccccc)OnHandleDestroyed*

19  
20 [C#] protected virtual void OnHandleDestroyed(EventArgs e);

21 [C++] protected: virtual void OnHandleDestroyed(EventArgs\* e);

22 [VB] Overridable Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

23 [JScript] protected function OnHandleDestroyed(e : EventArgs);

24  
25 *Description*

Raises the **System.Windows.Forms.Control.HandleDestroyed** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *ddddddddddd)OnHelpRequested*

[C#] protected virtual void OnHelpRequested(HelpEventArgs hevent);

[C++] protected: virtual void OnHelpRequested(HelpEventArgs\* hevent);

[VB] Overridable Protected Sub OnHelpRequested(ByVal hevent As HelpEventArgs)

[JScript] protected function OnHelpRequested(hevent : HelpEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.HelpRequested** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.HelpEventArgs** that contains the event data.

#### *eeeeeeeeeee)OnImeModeChanged*

[C#] protected virtual void OnImeModeChanged(EventArgs e);

[C++] protected: virtual void OnImeModeChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnImeModeChanged(ByVal e As EventArgs)

[JScript] protected function OnImeModeChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.ImeModeChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *ffffffffffff)OnInvalidated*

```
1  
2 [C#] protected virtual void OnInvalidated(InvalidateEventArgs e);  
3 [C++] protected: virtual void OnInvalidated(InvalidateEventArgs* e);  
4 [VB] Overridable Protected Sub OnInvalidated(ByVal e As InvalidateEventArgs)  
5 [JScript] protected function OnInvalidated(e : InvalidateEventArgs);  
6
```

#### *Description*

Raises the **System.Windows.Forms.Control.Invalidated** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.Windows.Forms.InvalidateEventArgs** that contains the event data.

### *gggggggggggg)OnKeyDown*

```
12  
13  
14 [C#] protected virtual void OnKeyDown(KeyEventArgs e);  
15 [C++] protected: virtual void OnKeyDown(KeyEventArgs* e);  
16 [VB] Overridable Protected Sub OnKeyDown(ByVal e As KeyEventArgs)  
17 [JScript] protected function OnKeyDown(e : KeyEventArgs);  
18
```

#### *Description*

Raises the **System.Windows.Forms.Control.KeyDown** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyEventArgs** that contains the event data.

### *hhhhhhhhhhhh)OnKeyPress*

```
23  
24  
25 [C#] protected virtual void OnKeyPress(KeyPressEventArgs e);
```

```

1 [C++] protected: virtual void OnKeyPress(KeyPressEventArgs* e);
2 [VB] Overridable Protected Sub OnKeyPress(ByVal e As KeyPressEventArgs)
3 [JScript] protected function OnKeyPress(e : KeyPressEventArgs);
4

```

#### *Description*

Raises the **System.Windows.Forms.Control.KeyPress** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyPressEventArgs** that contains the event data.

#### *iiiiiiiiii)OnKeyUp*

```

11 [C#] protected virtual void OnKeyUp(KeyEventArgs e);
12 [C++] protected: virtual void OnKeyUp(KeyEventArgs* e);
13 [VB] Overridable Protected Sub OnKeyUp(ByVal e As KeyEventArgs)
14 [JScript] protected function OnKeyUp(e : KeyEventArgs);
15

```

#### *Description*

Raises the **System.Windows.Forms.Control.KeyUp** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyEventArgs** that contains the event data.

#### *jjjjjjjjjj)OnLayout*

```

22 [C#] protected virtual void OnLayout(LayoutEventArgs levent);
23 [C++] protected: virtual void OnLayout(LayoutEventArgs* levent);
24 [VB] Overridable Protected Sub OnLayout(ByVal levent As LayoutEventArgs)
25

```

[JScript] protected function OnLayout(Levent : LayoutEventArgs);

### Description

Raises the **System.Windows.Forms.Control.Layout** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.LayoutEventArgs** that contains the event data.

*kkkkkkkkkkkk)OnLeave*

[C#] protected virtual void OnLeave(EventArgs e);

[C++] protected: virtual void OnLeave(EventArgs\* e);

[VB] Overridable Protected Sub OnLeave(ByVal e As EventArgs)

[JScript] protected function OnLeave(e : EventArgs);

### Description

Raises the **System.Windows.Forms.Control.Leave** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*llllllllll)OnLocationChanged*

[C#] protected virtual void OnLocationChanged(EventArgs e);

[C++] protected: virtual void OnLocationChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnLocationChanged(ByVal e As EventArgs)

[JScript] protected function OnLocationChanged(e : EventArgs);

### Description

Raises the **System.Windows.Forms.Control.LocationChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *OnLostFocus*

[C#]      protected      virtual      void      OnLostFocus(EventArgs      e);

[C++]      protected:      virtual      void      OnLostFocus(EventArgs\*      e);

[VB]      Overridable      Protected      Sub      OnLostFocus(ByVal e As EventArgs)

[JScript]      protected      function      OnLostFocus(e :      EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.LostFocus** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *OnMouseDown*

[C#]      protected      virtual      void      OnMouseDown(MouseEventArgs      e);

[C++]      protected:      virtual      void      OnMouseDown(MouseEventArgs\*      e);

[VB]      Overridable      Protected      Sub      OnMouseDown(ByVal e As MouseEventArgs)

[JScript]      protected      function      OnMouseDown(e :      MouseEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.MouseDown** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

## *oooooooooooo)OnMouseEnter*

```
[C#]      protected      virtual      void      OnMouseEnter(EventArgs      e);
[C++]      protected:      virtual      void      OnMouseEnter(EventArgs*      e);
[VB]      Overridable Protected Sub OnMouseEnter(ByVal e As EventArgs)
[JScript]      protected      function      OnMouseEnter(e      :      EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Control.MouseEnter** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## *oooooooooooo)OnMouseHover*

```
[C#]      protected      virtual      void      OnMouseHover(EventArgs      e);
[C++]      protected:      virtual      void      OnMouseHover(EventArgs*      e);
[VB]      Overridable Protected Sub OnMouseHover(ByVal e As EventArgs)
[JScript]      protected      function      OnMouseHover(e      :      EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Control.MouseHover** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## *oooooooooooo)OnMouseLeave*

```
[C#]      protected      virtual      void      OnMouseLeave(EventArgs      e);
[C++]      protected:      virtual      void      OnMouseLeave(EventArgs*      e);
```

1 [VB] Overridable Protected Sub OnMouseLeave(ByVal e As EventArgs)

2 [JScript] protected function OnMouseLeave(e : EventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.Control.MouseLeave** event.

6 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

7  
8 *rrrrrrrrrrrr)OnMouseMove*

9 [C#] protected virtual void OnMouseMove(MouseEventArgs e);

10 [C++] protected: virtual void OnMouseMove(MouseEventArgs\* e);

11 [VB] Overridable Protected Sub OnMouseMove(ByVal e As MouseEventArgs)

12 [JScript] protected function OnMouseMove(e : MouseEventArgs);

13  
14 *Description*

15 Raises the **System.Windows.Forms.Control.MouseMove** event.

16 Raising an event invokes the event handler through a delegate. For more  
information, see . A **System.Windows.Forms.MouseEventArgs** that contains  
the event data.

17  
18 *ssssssssss)OnMouseUp*

19  
20 [C#] protected virtual void OnMouseUp(MouseEventArgs e);

21 [C++] protected: virtual void OnMouseUp(MouseEventArgs\* e);

22 [VB] Overridable Protected Sub OnMouseUp(ByVal e As MouseEventArgs)

23 [JScript] protected function OnMouseUp(e : MouseEventArgs);



## Description

Raises the **System.Windows.Forms.Control.MouseUp** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

## tttttttttt)OnMouseWheel

[C#]      protected      virtual      void      OnMouseWheel(MouseEventArgs    e);

[C++]    protected:    virtual    void    OnMouseWheel(MouseEventArgs\*   e);

[VB]    Overridable Protected Sub OnMouseWheel(ByVal e As MouseEventArgs)

[JScript]    protected    function    OnMouseWheel(e    :    MouseEventArgs);

## Description

Raises the **System.Windows.Forms.Control.MouseWheel** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

## uuuuuuuuuuuu)OnMove

[C#]      protected      virtual      void      OnMove(EventArgs      e);

[C++]    protected:    virtual    void    OnMove(EventArgs\*    e);

[VB]    Overridable Protected Sub OnMove(ByVal e As EventArgs)

[JScript]    protected    function    OnMove(e      :      EventArgs);

## Description

Raises the **System.Windows.Forms.Control.Move** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *vvvvvvvvvvvv)OnNotifyMessage*

[C#]      protected      virtual      void      OnNotifyMessage(Message      m);

[C++]      protected:      virtual      void      OnNotifyMessage(Message      m);

[VB]      Overridable      Protected      Sub      OnNotifyMessage(ByVal m As Message)

[JScript]      protected      function      OnNotifyMessage(m      :      Message);

#### *Description*

Notifies the control of Windows messages.

The **System.Windows.Forms.Control.OnNotifyMessage(System.Windows.Forms.Message)** method is called if the control's **System.Windows.Forms.ControlStyles.EnableNotifyMessage** style bit is set. The **System.Windows.Forms.ControlStyles.EnableNotifyMessage** style allows the control to be notified when the **System.Windows.Forms.Control.WndProc(System.Windows.Forms.Message@)** method receives a Windows message. This method allows semi-trusted controls to listen for Windows messages without allowing them to modify the message. A **System.Windows.Forms.Message** that represents the Windows message.

#### *wwwwwwwwwwww)OnPaint*

[C#]      protected      virtual      void      OnPaint(PaintEventArgs      e);

[C++]      protected:      virtual      void      OnPaint(PaintEventArgs\*      e);

[VB]      Overridable      Protected      Sub      OnPaint(ByVal e As PaintEventArgs)

[JScript]      protected      function      OnPaint(e      :      PaintEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.Paint** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.PaintEventArgs** that contains the event data.

*xxxxxxxxxxx)OnPaintBackground*

[C#] protected virtual void OnPaintBackground(PaintEventArgs pevent);

[C++] protected: virtual void OnPaintBackground(PaintEventArgs\* pevent);

[VB] Overridable Protected Sub OnPaintBackground(ByVal pevent As PaintEventArgs)

[JScript] protected function OnPaintBackground(pevent : PaintEventArgs);

### *Description*

Paints the background of the control.

Inheriting classes should override this method to handle the erase background request from windows. When overriding

**System.Windows.Forms.Control.OnPaintBackground(System.Windows.Forms.PaintEventArgs)** in a derived class it is not necessary to call the base class's

**System.Windows.Forms.Control.OnPaintBackground(System.Windows.Forms.PaintEventArgs)** method. A

**System.Windows.Forms.PaintEventArgs** that contains information about the control to paint.

*yyyyyyyyyyyy)OnParentBackColorChanged*

[C#] protected virtual void OnParentBackColorChanged(EventArgs e);

[C++] protected: virtual void OnParentBackColorChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentBackColorChanged(ByVal e As EventArgs)



EventArgs)

[JScript] protected function OnParentBindingContextChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.BindingContextChanged** event when the **System.Windows.Forms.Control.BindingContext** property value of the control's container changes.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *bbbbbbbbbbbb)OnParentChanged*

[C#] protected virtual void OnParentChanged(EventArgs e);

[C++] protected: virtual void OnParentChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentChanged(ByVal e As EventArgs)

[JScript] protected function OnParentChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.ParentChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *cccccccccccc)OnParentEnabledChanged*

[C#] protected virtual void OnParentEnabledChanged(EventArgs e);

[C++] protected: virtual void OnParentEnabledChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentEnabledChanged(ByVal e As

EventArgs)

1 [JScript] protected function OnParentEnabledChanged(e : EventArgs);

3 *Description*

4 Raises the **System.Windows.Forms.Control.EnabledChanged** event when  
5 the **System.Windows.Forms.Control.Enabled** property value of the control's  
6 container changes.

7 Raising an event invokes the event handler through a delegate. For more  
8 information, see . An **System.EventArgs** that contains the event data.

9 *dddddddddddd)OnParentFontChanged*

10 [C#] protected virtual void OnParentFontChanged(EventArgs e);

11 [C++] protected: virtual void OnParentFontChanged(EventArgs\* e);

12 [VB] Overridable Protected Sub OnParentFontChanged(ByVal e As EventArgs)

13 [JScript] protected function OnParentFontChanged(e : EventArgs);

14 *Description*

15 Raises the **System.Windows.Forms.Control.FontChanged** event when the  
16 **System.Windows.Forms.Control.Font** property value of the control's  
17 container changes.

18 Raising an event invokes the event handler through a delegate. For more  
19 information, see . An **System.EventArgs** that contains the event data.

20 *eeeeeeeeeeee)OnParentForeColorChanged*

21 [C#] protected virtual void OnParentForeColorChanged(EventArgs e);

22 [C++] protected: virtual void OnParentForeColorChanged(EventArgs\* e);

23 [VB] Overridable Protected Sub OnParentForeColorChanged(ByVal e As  
24 EventArgs)

1 [JScript] protected function OnParentForeColorChanged(e : EventArgs);

2  
3 *Description*

4 Raises the **System.Windows.Forms.Control.ForeColorChanged** event when  
5 the **System.Windows.Forms.Control.ForeColor** property value of the  
6 control's container changes.

7 Raising an event invokes the event handler through a delegate. For more  
8 information, see . An **System.EventArgs** that contains the event data.

9  
10 *ffffffffffff)OnParentRightToLeftChanged*

11 [C#] protected virtual void OnParentRightToLeftChanged(EventArgs e);

12 [C++] protected: virtual void OnParentRightToLeftChanged(EventArgs\* e);

13 [VB] Overridable Protected Sub OnParentRightToLeftChanged(ByVal e As  
14 EventArgs)

15 [JScript] protected function OnParentRightToLeftChanged(e : EventArgs);

16  
17 *Description*

18 Raises the **System.Windows.Forms.Control.RightToLeftChanged** event  
19 when the **System.Windows.Forms.Control.RightToLeft** property value of  
20 the control's container changes.

21 Raising an event invokes the event handler through a delegate. For more  
22 information, see . An **System.EventArgs** that contains the event data.

23  
24 *gggggggggggg)OnParentVisibleChanged*

25 [C#] protected virtual void OnParentVisibleChanged(EventArgs e);

[C++] protected: virtual void OnParentVisibleChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentVisibleChanged(ByVal e As

EventArgs)

[JScript] protected function OnParentVisibleChanged(e : EventArgs);

#### Description

Raises the **System.Windows.Forms.Control.VisibleChanged** event when the **System.Windows.Forms.Control.Visible** property value of the control's container changes.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### hhhhhhhhhhhh)OnQueryContinueDrag

[C#] protected virtual void OnQueryContinueDrag(QueryContinueDragEventArgs qcdevent);

[C++] protected: virtual void OnQueryContinueDrag(QueryContinueDragEventArgs\* qcdevent);

[VB] Overridable Protected Sub OnQueryContinueDrag(ByVal qcdevent As QueryContinueDragEventArgs)

[JScript] protected function OnQueryContinueDrag(qcdevent : QueryContinueDragEventArgs);

#### Description

Raises the **System.Windows.Forms.Control.QueryContinueDrag** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

**System.Windows.Forms.QueryContinueDragEventArgs** that contains the event data.



### iiiiiiiiiii)OnResize

```
[C#]      protected      virtual      void      OnResize(EventArgs      e);
[C++]      protected:      virtual      void      OnResize(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnResize(ByVal e As EventArgs)
[JScript]      protected      function      OnResize(e      :      EventArgs);
```

#### Description

Raises the **System.Windows.Forms.Control.Resize** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### iiiiiiiiiii)OnRightToLeftChanged

```
[C#]      protected      virtual      void      OnRightToLeftChanged(EventArgs      e);
[C++]      protected:      virtual      void      OnRightToLeftChanged(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnRightToLeftChanged(ByVal e As EventArgs)
[JScript]      protected      function      OnRightToLeftChanged(e      :      EventArgs);
```

#### Description

Raises the **System.Windows.Forms.Control.RightToLeftChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### kkkkkkkkkkkkk)OnSizeChanged

```
[C#]      protected      virtual      void      OnSizeChanged(EventArgs      e);
[C++]      protected:      virtual      void      OnSizeChanged(EventArgs*      e);
```

1 [VB] Overridable Protected Sub OnSizeChanged(ByVal e As EventArgs)

2 [JScript] protected function OnSizeChanged(e : EventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.Control.SizeChanged** event.

6 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

7  
8 *llllllllll)OnStyleChanged*

9 [C#] protected virtual void OnStyleChanged(EventArgs e);

10 [C++] protected: virtual void OnStyleChanged(EventArgs\* e);

11 [VB] Overridable Protected Sub OnStyleChanged(ByVal e As EventArgs)

12 [JScript] protected function OnStyleChanged(e : EventArgs);

13  
14 *Description*

15 Raises the **System.Windows.Forms.Control.StyleChanged** event.

16 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

17  
18 *mmmmmmmmmmmm)OnSystemColorsChanged*

19  
20 [C#] protected virtual void OnSystemColorsChanged(EventArgs e);

21 [C++] protected: virtual void OnSystemColorsChanged(EventArgs\* e);

22 [VB] Overridable Protected Sub OnSystemColorsChanged(ByVal e As  
23 EventArgs)

24 [JScript] protected function OnSystemColorsChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.Control.SystemColorsChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*nnnnnnnnnnnnnn)OnTabIndexChanged*

[C#]   protected   virtual   void   OnTabIndexChanged(EventArgs   e);

[C++]   protected:   virtual   void   OnTabIndexChanged(EventArgs\*   e);

[VB]   Overridable Protected Sub OnTabIndexChanged(ByVal e As EventArgs)

[JScript]   protected   function   OnTabIndexChanged(e   :   EventArgs);

*Description*

Raises the **System.Windows.Forms.Control.TabIndexChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*ooooooooooooo)OnTabStopChanged*

[C#]   protected   virtual   void   OnTabStopChanged(EventArgs   e);

[C++]   protected:   virtual   void   OnTabStopChanged(EventArgs\*   e);

[VB]   Overridable Protected Sub OnTabStopChanged(ByVal e As EventArgs)

[JScript]   protected   function   OnTabStopChanged(e   :   EventArgs);

*Description*

Raises the **System.Windows.Forms.Control.TabStopChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *pppppppppppppp)OnTextChanged*

[C#]      protected      virtual      void      OnTextChanged(EventArgs    e);

[C++]      protected:      virtual      void      OnTextChanged(EventArgs\*    e);

[VB]    Overridable   Protected   Sub   OnTextChanged(ByVal   e   As   EventArgs)

[JScript]      protected      function      OnTextChanged(e      :      EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.TextChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *qqqqqqqqqqqqq)OnValidated*

[C#]      protected      virtual      void      OnValidated(EventArgs    e);

[C++]      protected:      virtual      void      OnValidated(EventArgs\*    e);

[VB]    Overridable   Protected   Sub   OnValidated(ByVal   e   As   EventArgs)

[JScript]      protected      function      OnValidated(e      :      EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.Validated** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *rrrrrrrrrrrr)OnValidating*

```
[C#]    protected    virtual    void    OnValidating(CancelEventArgs    e);
[C++]    protected:    virtual    void    OnValidating(CancelEventArgs*    e);
[VB]    Overridable Protected Sub OnValidating(ByVal e As CancelEventArgs)
[JScript]    protected    function    OnValidating(e    :    CancelEventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.Validating** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.ComponentModel.CancelEventArgs** that contains the event data.

### *ssssssssssss)OnVisibleChanged*

```
[C#]    protected    virtual    void    OnVisibleChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnVisibleChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnVisibleChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnVisibleChanged(e    :    EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.VisibleChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *tttttttttt)PerformLayout*

```
[C#]                public                void                PerformLayout();
```



```

1 [VB] Public Function PointToClient(ByVal p As Point) As Point
2 [JScript] public function PointToClient(p : Point) : Point;

```

#### *Description*

Computes the location of the specified screen point to client coordinates.  
*Return Value:* A **System.Drawing.Point** that represents the converted **System.Drawing.Point**,  $p$ , in client coordinates. The screen coordinate **System.Drawing.Point** to convert.

*xxxxxxxxxxxxxxxxx)PointToScreen*

```

9 [C#] public Point PointToScreen(Point p);
10 [C++] public: Point PointToScreen(Point p);
11 [VB] Public Function PointToScreen(ByVal p As Point) As Point
12 [JScript] public function PointToScreen(p : Point) : Point;

```

#### *Description*

Computes the location of the specified client point to screen coordinates.  
*Return Value:* A **System.Drawing.Point** that represents the converted **System.Drawing.Point**,  $p$ , in screen coordinates. The screen coordinate **System.Drawing.Point** to convert.

*xxxxxxxxxxxxxxxxx)PreProcessMessage*

```

20 [C#] public virtual bool PreProcessMessage(ref Message msg);
21 [C++] public: virtual bool PreProcessMessage(Message* msg);
22 [VB] Overridable Public Function PreProcessMessage(ByRef msg As Message)
23 As Boolean
24 [JScript] public function PreProcessMessage(msg : Message) : Boolean;

```

## Description

Preprocesses input messages within the message loop before they are dispatched.

*Return Value:* **true** if the message was processed by the control; otherwise, **false**.

This method is only called when the control is hosted inside of a Windows Forms application or as an ActiveX control. A **System.Windows.Forms.Message**, passed by reference, that represents the message to process.

~~~~~*ProcessCmdKey*

[C#] protected virtual bool ProcessCmdKey(ref Message msg, Keys keyData);

[C++] protected: virtual bool ProcessCmdKey(Message\* msg, Keys keyData);

[VB] Overridable Protected Function ProcessCmdKey(ByRef msg As Message,

ByVal keyData As Keys) As Boolean

[JScript] protected function ProcessCmdKey(msg : Message, keyData : Keys) : Boolean;

## Description

Processes a command key.

*Return Value:* **true** if the character was processed by the control; otherwise, **false**.

This method is called during message preprocessing to handle command keys. Command keys are keys that always take precedence over regular input keys. Examples of command keys include accelerators and menu shortcuts. The method must return **true** to indicate that it has processed the command key, or **false** to indicate that the key is not a command key. This method is only called when the control is hosted inside of a Windows Forms application or as an ActiveX control. A **System.Windows.Forms.Message**, passed by reference, that represents the window message to process. The key code for the key the user presses.





1  
2 *Description*

3 Processes a dialog key.

4 *Return Value:* **true** if the key was processed by the control; otherwise, **false** .

5 This method is called during message preprocessing to handle dialog characters,  
6 such as TAB, RETURN, ESCAPE, and arrow keys. This method is called only if the  
7 **System.Windows.Forms.Control.IsInputKey(System.Windows.Forms.Keys)**  
8 **System.Windows.Forms.Control.ProcessDialogKey(System.Windows.Forms.Keys)** method indicates that the control is not processing the key. The  
9 **System.Windows.Forms.Control.ProcessDialogKey(System.Windows.Forms.Keys)** simply sends the character to the parent's  
10 **System.Windows.Forms.Control.ProcessDialogKey(System.Windows.Forms.Keys)** method, or returns **false** if the control has no parent. The  
11 **System.Windows.Forms.Form** class overrides this method to perform actual  
12 processing of dialog keys. This method is only called when the control is hosted  
13 inside of a Windows Forms application or as an ActiveX control. One of the  
14 **System.Windows.Forms.Keys** values that represents the key to process.

15 *bbbbbbbbbbbbbbbb)ProcessEventArgs*

16 [C#] protected virtual bool ProcessEventArgs(ref Message m);

17 [C++] protected: virtual bool ProcessEventArgs(Message\* m);

18 [VB] Overridable Protected Function ProcessEventArgs(ByRef m As  
19 Message) As Boolean

20 [JScript] protected function ProcessEventArgs(m : Message) : Boolean;

21  
22 *Description*

23 Processes a key message and generates the appropriate control events.

24 *Return Value:* **true** if the message was processed by the control; otherwise,  
25 **false** .

26 This method is called when a control receives a keyboard message. The method  
27 is responsible for generating the appropriate key events for the message by  
28 calling the  
29 **System.Windows.Forms.Control.OnKeyPress(System.Windows.Forms.KeyPressEventArgs)** ,

**System.Windows.Forms.Control.OnKeyDown(System.Windows.Forms.KeyEventArgs)** , or  
**System.Windows.Forms.Control.OnKeyUp(System.Windows.Forms.KeyEventArgs)** methods. The *m* parameter contains the window message that must be processed. Possible values for the **System.Windows.Forms.Message.Msg** property are WM\_CHAR, WM\_KEYDOWN, WM\_SYSKEYDOWN, WM\_KEYUP, WM\_SYSKEYUP, and WM\_IMECHAR. A **System.Windows.Forms.Message**, passed by reference, that represents the window message to process.

*cccccccccccccc)ProcessKeyMessage*

[C#] protected internal virtual bool ProcessKeyMessage(ref Message m);

[C++] protected public: virtual bool ProcessKeyMessage(Message\* m);

[VB] Overridable Protected Friend Dim Function ProcessKeyMessage(ByRef m

As Message) As Boolean

[JScript] package function ProcessKeyMessage(m : Message) : Boolean;

### *Description*

Processes a keyboard message.

*Return Value:* **true** if the message was processed by the control; otherwise, **false** .

This method is called when a control receives a keyboard message. The method first checks if the control has a parent, and if so calls the parent's **System.Windows.Forms.Control.ProcessKeyMessage(System.Windows.Forms.Message@)** method. If the parent's **System.Windows.Forms.Control.ProcessKeyMessage(System.Windows.Forms.Message@)** method does not process the message then the **System.Windows.Forms.Control.ProcessKeyEventArgs(System.Windows.Forms.Message@)** method is called to generate the appropriate keyboard events. The *m* parameter contains the window message that must be processed. Possible values for the **System.Windows.Forms.Message.Msg** property are WM\_CHAR, WM\_KEYDOWN, WM\_SYSKEYDOWN, WM\_KEYUP, and WM\_SYSKEYUP. A **System.Windows.Forms.Message**, passed by reference, that represents the window message to process.

## *dddddddddddddd)ProcessKeyPreview*

```
1 [C#]    protected    virtual    bool    ProcessKeyPreview(ref    Message    m);
2
3 [C++]    protected:    virtual    bool    ProcessKeyPreview(Message*    m);
4
5 [VB] Overridable Protected Function ProcessKeyPreview(ByRef m As Message)
6
7 As Boolean
8
9 [JScript] protected function ProcessKeyPreview(m : Message) : Boolean;
```

### *Description*

Previews a keyboard message.

*Return Value:* **true** if the message was processed by the control; otherwise, **false**.

This method is called by a child control when the child control receives a keyboard message. The child control calls this method before generating any keyboard events for the message. If this method returns **true**, the child control considers the message processed and does not generate any keyboard events. The *m* parameter contains the window message to preview. Possible values for the **System.Windows.Forms.Message.Msg** property are WM\_CHAR, WM\_KEYDOWN, WM\_SYSKEYDOWN, WM\_KEYUP, and WM\_SYSKEYUP. The **System.Windows.Forms.Control.ProcessKeyPreview(System.Windows.Forms.Message@)** method simply sends the character to the parent's **System.Windows.Forms.Control.ProcessKeyPreview(System.Windows.Forms.Message@)** method, or returns **false** if the control has no parent. The **System.Windows.Forms.Form** class overrides this method to perform actual processing of dialog keys. A **System.Windows.Forms.Message**, passed by reference, that represents the window message to process.

## *eeeeeeeeeeeeeeee)ProcessMnemonic*

```
21
22 [C#]    protected    virtual    bool    ProcessMnemonic(char    charCode);
23
24 [C++]    protected:    virtual    bool    ProcessMnemonic(__wchar_t    charCode);
25
26 [VB] Overridable Protected Function ProcessMnemonic(ByVal charCode As
27 Char) As Boolean
```

[JScript] protected function ProcessMnemonic(charCode : Char) : Boolean;

### *Description*

Processes a mnemonic character.

*Return Value:* **true** if the character was processed as a mnemonic by the control; otherwise, **false** .

This method is called to give a control the opportunity to process a mnemonic character. The method should check if the control is in a state to process mnemonics and if the given character represents a mnemonic. If so, the method should perform the action associated with the mnemonic and return **true** . If not, the method should return **false** . Implementations of this method often use the

**System.Windows.Forms.Control.IsMnemonic(System.Char, System.String)** method to check if the given character matches a mnemonic in the control's text, for example: if (CanSelect && IsMnemonic(charCode, MyControl.Text) { // perform action associated with mnemonic } This default implementation of the **System.Windows.Forms.Control.ProcessMnemonic(System.Char)** method simply returns **false** to indicate that the control has no mnemonic. The character to process.

### *ffffffffffff)RaiseDragEvent*

[C#] protected void RaiseDragEvent(object key, DragEventArgs e);

[C++] protected: void RaiseDragEvent(Object\* key, DragEventArgs\* e);

[VB] Protected Sub RaiseDragEvent(ByVal key As Object, ByVal e As DragEventArgs)

[JScript] protected function RaiseDragEvent(key : Object, e : DragEventArgs);

### *Description*

Raises the the appropriate drag event. The event to raise. A

**System.Windows.Forms.DragEventArgs** that contains the event data.

## *gggggggggggggg)RaiseKeyEvent*

```
1 [C#] protected void RaiseKeyEvent(object key, EventArgs e);
2
3 [C++] protected: void RaiseKeyEvent(Object* key, EventArgs* e);
4
5 [VB] Protected Sub RaiseKeyEvent(ByVal key As Object, ByVal e As
6   EventArgs)
7
8 [JScript] protected function RaiseKeyEvent(key : Object, e : EventArgs);
```

### *Description*

Raises the the appropriate key event. The event to raise. A **System.Windows.Forms.KeyEventArgs** that contains the event data.

## *hhhhhhhhhhhhhh)RaiseMouseEvent*

```
13 [C#] protected void RaiseMouseEvent(object key, MouseEventArgs e);
14
15 [C++] protected: void RaiseMouseEvent(Object* key, MouseEventArgs* e);
16
17 [VB] Protected Sub RaiseMouseEvent(ByVal key As Object, ByVal e As
18   MouseEventArgs)
19
20 [JScript] protected function RaiseMouseEvent(key : Object, e : MouseEventArgs);
```

### *Description*

Raises the the appropriate mouse event. The event to raise. A **System.Windows.Forms.MouseEventArgs** that contains the event data.

## *iiiiiiiiiii)RaisePaintEvent*

```
24 [C#] protected void RaisePaintEvent(object key, PaintEventArgs e);
25
26 [C++] protected: void RaisePaintEvent(Object* key, PaintEventArgs* e);
```

[VB] Protected Sub RaisePaintEvent(ByVal key As Object, ByVal e As PaintEventArgs)

[JScript] protected function RaisePaintEvent(key : Object, e : PaintEventArgs);

#### Description

Raises the the appropriate paint event. The event to raise. A **System.Windows.Forms.PaintEventArgs** that contains the event data.

#### *jjjjjjjjjjjj)RecreateHandle*

|           |            |          |                   |
|-----------|------------|----------|-------------------|
| [C#]      | protected  | void     | RecreateHandle(); |
| [C++]     | protected: | void     | RecreateHandle(); |
| [VB]      | Protected  | Sub      | RecreateHandle()  |
| [JScript] | protected  | function | RecreateHandle(); |

#### Description

Forces the re-creation of the handle for the control.

When overriding this method, you must call the base class implementation of this method.

#### *kkkkkkkkkkkkkk)RectangleToClient*

|           |                 |                                         |                                               |
|-----------|-----------------|-----------------------------------------|-----------------------------------------------|
| [C#]      | public          | Rectangle                               | RectangleToClient(Rectangle r);               |
| [C++]     | public:         | Rectangle                               | RectangleToClient(Rectangle r);               |
| [VB]      | Public Function | RectangleToClient(ByVal r As Rectangle) | As Rectangle                                  |
| [JScript] | public          | function                                | RectangleToClient(r : Rectangle) : Rectangle; |

#### Description

Computes the size and location of the specified screen rectangle to client coordinates.

*Return Value:* A **System.Drawing.Rectangle** that represents the converted **System.Drawing.Rectangle**, *r*, in client coordinates. The screen coordinate **System.Drawing.Rectangle** to convert.

#### *lllllllllll)RectangleToScreen*

[C#]        public        Rectangle        RectangleToScreen(Rectangle        r);

[C++]        public:        Rectangle        RectangleToScreen(Rectangle        r);

[VB] Public Function RectangleToScreen(ByVal r As Rectangle) As Rectangle

[JScript] public function RectangleToScreen(r : Rectangle) : Rectangle;

#### *Description*

Computes the size and location of the specified client rectangle to screen coordinates.

*Return Value:* A **System.Drawing.Rectangle** that represents the converted **System.Drawing.Rectangle**, *p*, in screen coordinates. The screen coordinate **System.Drawing.Rectangle** to convert.

#### *mmmmmmmmmmmmmmmm)ReflectMessage*

[C#] protected static bool ReflectMessage(IntPtr hWnd, ref Message m);

[C++] protected: static bool ReflectMessage(IntPtr hWnd, Message\* m);

[VB] Protected Shared Function ReflectMessage(ByVal hWnd As IntPtr, ByRef m

As                      Message) As                      Boolean

[JScript] protected static function ReflectMessage(hWnd : IntPtr, m : Message) :

Boolean;

#### *Description*





You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control** .

*ResetBindings*

|           |         |          |                  |
|-----------|---------|----------|------------------|
| [C#]      | public  | void     | ResetBindings(); |
| [C++]     | public: | void     | ResetBindings(); |
| [VB]      | Public  | Sub      | ResetBindings()  |
| [JScript] | public  | function | ResetBindings(); |

*Description*

Resets the **System.Windows.Forms.Control.DataBindings** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control** .

*ResetCursor*

|           |             |          |      |                |
|-----------|-------------|----------|------|----------------|
| [C#]      | public      | virtual  | void | ResetCursor(); |
| [C++]     | public:     | virtual  | void | ResetCursor(); |
| [VB]      | Overridable | Public   | Sub  | ResetCursor()  |
| [JScript] | public      | function |      | ResetCursor(); |

*Description*

Resets the **System.Windows.Forms.Control.Cursor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control** .

## rrrrrrrrrrrrrr)ResetFont

|           |             |          |      |              |
|-----------|-------------|----------|------|--------------|
| [C#]      | public      | virtual  | void | ResetFont(); |
| [C++]     | public:     | virtual  | void | ResetFont(); |
| [VB]      | Overridable | Public   | Sub  | ResetFont()  |
| [JScript] | public      | function |      | ResetFont(); |

### Description

Resets the **System.Windows.Forms.Control.Font** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control**.

## ssssssssssssss)ResetForeColor

|           |             |          |      |                   |
|-----------|-------------|----------|------|-------------------|
| [C#]      | public      | virtual  | void | ResetForeColor(); |
| [C++]     | public:     | virtual  | void | ResetForeColor(); |
| [VB]      | Overridable | Public   | Sub  | ResetForeColor()  |
| [JScript] | public      | function |      | ResetForeColor(); |

### Description

Resets the **System.Windows.Forms.Control.ForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control**.

#### ResetImeMode

|           |         |          |                 |
|-----------|---------|----------|-----------------|
| [C#]      | public  | void     | ResetImeMode(); |
| [C++]     | public: | void     | ResetImeMode(); |
| [VB]      | Public  | Sub      | ResetImeMode()  |
| [JScript] | public  | function | ResetImeMode(); |

#### Description

Resets the **System.Windows.Forms.Control.ImeMode** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.Control** or creating your own control incorporating the **System.Windows.Forms.Control**.

#### ResetMouseEventArgs

|           |            |          |                        |
|-----------|------------|----------|------------------------|
| [C#]      | protected  | void     | ResetMouseEventArgs(); |
| [C++]     | protected: | void     | ResetMouseEventArgs(); |
| [VB]      | Protected  | Sub      | ResetMouseEventArgs()  |
| [JScript] | protected  | function | ResetMouseEventArgs(); |

#### Description

Resets the mouse leave listeners.

#### ResetRightToLeft

|       |         |         |      |                     |
|-------|---------|---------|------|---------------------|
| [C#]  | public  | virtual | void | ResetRightToLeft(); |
| [C++] | public: | virtual | void | ResetRightToLeft(); |



1 [JScript] public function ResumeLayout(); Resumes normal layout logic.

3 *Description*

4 Resumes normal layout logic.

5 Calling the **System.Windows.Forms.Control.ResumeLayout** method forces  
an immediate layout if there are any pending layout requests.

6 *yyyyyyyyyyyyyyyy)ResumeLayout*

8 [C#] public void ResumeLayout(bool performLayout);

9 [C++] public: void ResumeLayout(bool performLayout);

10 [VB] Public Sub ResumeLayout(ByVal performLayout As Boolean)

11 [JScript] public function ResumeLayout(performLayout : Boolean);

13 *Description*

14 Resumes normal layout logic. Optionally forces an immediate layout of pending  
15 layout requests.

16 Calling the **System.Windows.Forms.Control.ResumeLayout** method forces  
an immediate layout if there are any pending layout requests. When the  
17 *performLayout* parameter is set to **true** , an immediate layout occurs if there are  
any pending layout requests. **true** to execute pending layout requests;  
18 otherwise, **false**.

19 *zzzzzzzzzzzzzzzz)RtlTranslateAlignment*

21 [C#] protected ContentAlignment RtlTranslateAlignment(ContentAlignment  
22 align);

23 [C++] protected: ContentAlignment RtlTranslateAlignment(ContentAlignment  
24 align);

```

1 [VB] Protected Function RtlTranslateAlignment(ByVal align As
2 ContentAlignment) As ContentAlignment

```

```

3 [JScript] protected function RtlTranslateAlignment(align : ContentAlignment) :
4 ContentAlignment;

```

#### 6 *Description*

7 Converts the specified **System.Drawing.ContentAlignment** to the  
 8 appropriate **System.Drawing.ContentAlignment** to support right to left text.  
*Return Value:* One of the **System.Drawing.ContentAlignment** values. One of  
 9 the **System.Drawing.ContentAlignment** values.

10 *aaaaaaaaaaaaaaaa)RtlTranslateAlignment*

```

11
12 [C#] protected HorizontalAlignment RtlTranslateAlignment(HorizontalAlignment
13 align);

```

```

14 [C++] protected: HorizontalAlignment
15 RtlTranslateAlignment(HorizontalAlignment align);

```

```

16 [VB] Protected Function RtlTranslateAlignment(ByVal align As
17 HorizontalAlignment) As HorizontalAlignment

```

```

18 [JScript] protected function RtlTranslateAlignment(align : HorizontalAlignment) :
19 HorizontalAlignment; Converts the current alignment to the appropriate alignment
20 to support right to left text.

```

#### 21 *Description*

22 Converts the specified **System.Windows.Forms.HorizontalAlignment** to the  
 23 appropriate **System.Windows.Forms.HorizontalAlignment** to support right  
 24 to left text.

25 *Return Value:* One of the **System.Windows.Forms.HorizontalAlignment**  
 values. One of the **System.Windows.Forms.HorizontalAlignment** values.

### *bbbbbbbbbbbbbbbb)RtlTranslateAlignment*

[C#] protected LeftRightAlignment RtlTranslateAlignment(LeftRightAlignment align);

[C++] protected: LeftRightAlignment RtlTranslateAlignment(LeftRightAlignment align);

[VB] Protected Function RtlTranslateAlignment(ByVal align As LeftRightAlignment) As LeftRightAlignment

[JScript] protected function RtlTranslateAlignment(align : LeftRightAlignment) : LeftRightAlignment;

#### *Description*

Converts the specified **System.Windows.Forms.LeftRightAlignment** to the appropriate **System.Windows.Forms.LeftRightAlignment** to support right to left text.

*Return Value:* One of the **System.Windows.Forms.LeftRightAlignment** values. One of the **System.Windows.Forms.LeftRightAlignment** values.

### *cccccccccccccccc)RtlTranslateContent*

[C#] protected ContentAlignment RtlTranslateContent(ContentAlignment align);

[C++] protected: ContentAlignment RtlTranslateContent(ContentAlignment align);

[VB] Protected Function RtlTranslateContent(ByVal align As ContentAlignment) As ContentAlignment

[JScript] protected function RtlTranslateContent(align : ContentAlignment) : ContentAlignment;



## Description

Converts the specified **System.Drawing.ContentAlignment** to the appropriate **System.Drawing.ContentAlignment** to support right to left text.  
*Return Value:* One of the **System.Drawing.ContentAlignment** values. One of the **System.Drawing.ContentAlignment** values.

*ddddddddddddddd)RtlTranslateHorizontal*

[C#] protected HorizontalAlignment RtlTranslateHorizontal(HorizontalAlignment align);

[C++] protected: HorizontalAlignment  
RtlTranslateHorizontal(HorizontalAlignment align);

[VB] Protected Function RtlTranslateHorizontal(ByVal align As HorizontalAlignment) As HorizontalAlignment

[JScript] protected function RtlTranslateHorizontal(align : HorizontalAlignment) : HorizontalAlignment;

## Description

Converts the specified **System.Windows.Forms.HorizontalAlignment** to the appropriate **System.Windows.Forms.HorizontalAlignment** to support right to left text.

*Return Value:* One of the **System.Windows.Forms.HorizontalAlignment** values. One of the **System.Windows.Forms.HorizontalAlignment** values.

*eeeeeeeeeeeeeee)RtlTranslateLeftRight*

[C#] protected LeftRightAlignment RtlTranslateLeftRight(LeftRightAlignment align);

[C++] protected: LeftRightAlignment RtlTranslateLeftRight(LeftRightAlignment

align);

[VB] Protected Function RtlTranslateLeftRight(ByVal align As  
LeftRightAlignment) As LeftRightAlignment  
[JScript] protected function RtlTranslateLeftRight(align : LeftRightAlignment) :  
LeftRightAlignment;

#### *Description*

Converts the specified **System.Windows.Forms.LeftRightAlignment** to the appropriate **System.Windows.Forms.LeftRightAlignment** to support right to left text.

*Return Value:* One of the **System.Windows.Forms.LeftRightAlignment** values. One of the **System.Windows.Forms.LeftRightAlignment** values.

#### *Scale*

[C#] public void Scale(float ratio);  
[C++] public: void Scale(float ratio);  
[VB] Public Sub Scale(ByVal ratio As Single)  
[JScript] public function Scale(ratio : float); Scales the entire control and any child  
controls.

#### *Description*

Scales the entire control and any child controls to the specified ratio horizontally and vertically. The ratio by which to scale the control horizontally and vertically.

#### *Scale*

[C#] public void Scale(float dx, float dy);  
[C++] public: void Scale(float dx, float dy);

1 [VB] Public Sub Scale(ByVal dx As Single, ByVal dy As Single)

2 [JScript] public function Scale(dx : float, dy : float);

3  
4 *Description*

5 Scales the control and any child controls to the specified horizontal ratio and  
6 vertical ratio. The ratio by which to scale the control horizontally. The ratio by  
7 which to scale the control vertically.

8  
9 *hhhhhhhhhhhhhh)ScaleCore*

10 [C#] protected virtual void ScaleCore(float dx, float dy);

11 [C++] protected: virtual void ScaleCore(float dx, float dy);

12 [VB] Overridable Protected Sub ScaleCore(ByVal dx As Single, ByVal dy As  
13 Single)

14 [JScript] protected function ScaleCore(dx : float, dy : float);

15 *Description*

16 Performs the work of scaling the entire control and any child controls. Ratio to  
17 scale the control horizontally. Ratio to scale the control vertically.

18 *iiiiiiiiiii)Select*

19  
20 [C#] public void Select();

21 [C++] public: void Select();

22 [VB] Public Sub Select()

23 [JScript] public function Select(); Activates a control.

24  
25 *Description*

Activates this control.

The **System.Windows.Forms.Control.Select** method activates the control if the control has the **System.Windows.Forms.ControlStyles.Selectable** style bit set to **true** , is contained in another control, and all of its parent controls are both visible and enabled.

*iiiiiiiiiii)Select*

[C#] protected virtual void Select(bool directed, bool forward);

[C++] protected: virtual void Select(bool directed, bool forward);

[VB] Overridable Protected Sub Select(ByVal directed As Boolean, ByVal forward As Boolean)

[JScript] protected function Select(directed : Boolean, forward : Boolean);

#### *Description*

Activates a child control. Optionally specifies the direction in the tab order to select the control from.

The *directed* and *forward* parameters are used by container-style controls. When the *directed* parameter is set to **true** , the *forward* parameter is evaluated to determine which control to select. When *forward* is set to **true** the next control in the tab order is selected, when **false** , the previous control in the tab order is selected. **true** to specify the direction of the control to select; otherwise, **false**. **true** to select the next control forward in the tab order; otherwise, **false**.

*kkkkkkkkkkkkkk)SelectNextControl*

[C#] public bool SelectNextControl(Control ctl, bool forward, bool tabStopOnly,

bool nested, bool wrap);

[C++] public: bool SelectNextControl(Control\* ctl, bool forward, bool tabStopOnly, bool nested, bool wrap);

[VB] Public Function SelectNextControl(ByVal ctl As Control, ByVal forward As

```

1 Boolean, ByVal tabStopOnly As Boolean, ByVal nested As Boolean, ByVal wrap
2 As Boolean) As Boolean
3 [JScript] public function SelectNextControl(ctl : Control, forward : Boolean,
4 tabStopOnly : Boolean, nested : Boolean, wrap : Boolean) : Boolean;
5

```

#### *Description*

Activates the next control.

*Return Value:* **true** if a control was activated; otherwise, **false** .

This activates the control if the control has **System.Windows.Forms.ControlStyles.Selectable** set to **true** , is contained in another control, and all of its parent controls are both visible and enabled.

Examples of controls that are not selectable are

**System.Windows.Forms.Label** , **System.Windows.Forms.Panel** ,

**System.Windows.Forms.PictureBox** , and

**System.Windows.Forms.GroupBox** . The

**System.Windows.Forms.Control** where to start the search. **true** to select the next control forward in the tab order; otherwise, **false** . **true** to ignore the controls with the **System.Windows.Forms.Control.TabStop** property set to **false**; otherwise, **false** . **true** to include nested (children of child controls) child controls; otherwise, **false** . **true** to start searching from the first control in the tab order after the last control has been reached; otherwise, **false** .

#### *SendToBack*

|           |         |          |               |
|-----------|---------|----------|---------------|
| [C#]      | public  | void     | SendToBack(); |
| [C++]     | public: | void     | SendToBack(); |
| [VB]      | Public  | Sub      | SendToBack()  |
| [JScript] | public  | function | SendToBack(); |

#### *Description*

Sends the control to the back of the z-order.

The control is moved to the bottom of the z-order. If the control is a child of another control, the child control is moved to the bottom of the z-order.

#### *mmmmmmmmmmmmmmmmmmmm)SetBounds*

[C#] public void SetBounds(int x, int y, int width, int height);

[C++] public: void SetBounds(int x, int y, int width, int height);

[VB] Public Sub SetBounds(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer)

[JScript] public function SetBounds(x : int, y : int, width : int, height : int); Sets the bounds of the control.

#### *Description*

Sets the bounds of the control to the specified location and size. The new **System.Windows.Forms.Control.Left** property value of the control. The new **System.Windows.Forms.Control.Right** property value of the control. The new **System.Windows.Forms.Control.Width** property value of the control. The new **System.Windows.Forms.Control.Height** property value of the control.

#### *nnnnnnnnnnnnnnnnnnnn)SetBounds*

[C#] public void SetBounds(int x, int y, int width, int height, BoundsSpecified specified);

[C++] public: void SetBounds(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Public Sub SetBounds(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] public function SetBounds(x : int, y : int, width : int, height : int,

specified : BoundsSpecified);

### Description

Sets the specified bounds of the control to the specified location and size. The new **System.Windows.Forms.Control.Left** property value of the control. The new **System.Windows.Forms.Control.Right** property value of the control. The new **System.Windows.Forms.Control.Width** property value of the control. The new **System.Windows.Forms.Control.Height** property value of the control. A bitwise combination of the **System.Windows.Forms.BoundsSpecified** values. For any parameter not specified, the current value will be used.

*ooooooooooooooooo)SetBoundsCore*

[C#] protected virtual void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: virtual void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overridable Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

### Description

Performs the work of setting the specified bounds of this control.

Typically, the parameters that correspond to the bounds not included in the *specified* parameter are passed in with their current values. For example, the **System.Windows.Forms.Control.Height** , **System.Windows.Forms.Control.Width** , or the **System.Drawing.Point.X** or **System.Drawing.Point.Y** properties of the

**System.Windows.Forms.Control.Location** property can be passed in with a reference to the current instance of the control. The new **System.Windows.Forms.Control.Left** property value of the control. The new **System.Windows.Forms.Control.Right** property value of the control. The new **System.Windows.Forms.Control.Width** property value of the control. The new **System.Windows.Forms.Control.Height** property value of the control. A bitwise combination of the **System.Windows.Forms.BoundsSpecified** values.

*pppppppppppppppp)SetClientSizeCore*

[C#]     protected     virtual     void     SetClientSizeCore(int     x,     int     y);

[C++]     protected:     virtual     void     SetClientSizeCore(int     x,     int     y);

[VB]     Overridable Protected Sub SetClientSizeCore(ByVal x As Integer, ByVal y As Integer)

[JScript]     protected     function     SetClientSizeCore(x     :     int,     y     :     int);

#### *Description*

Sets the size of the client area of the control.

The client area starts at (0, 0) location and extends to the ( *x* , *y* ) location. The client area width, in pixels. The client area height, in pixels.

*qqqqqqqqqqqqqqqq)SetStyle*

[C#]     protected     void     SetStyle(ControlStyles     flag,     bool     value);

[C++]     protected:     void     SetStyle(ControlStyles     flag,     bool     value);

[VB]     Protected Sub SetStyle(ByVal flag As ControlStyles, ByVal value As Boolean)

[JScript]     protected     function     SetStyle(flag     :     ControlStyles,     value     :     Boolean);



### Description

Sets the specified style bit to the specified value.

Control style bit flags are used to categorize supported behavior. A control can enable a style by calling the **System.Windows.Forms.Control.SetStyle(System.Windows.Forms.ControlStyles, System.Boolean)** method and passing in the appropriate **System.Windows.Forms.ControlStyles** bit and the boolean value set the bit to. To determine the value assigned to a specified **System.Windows.Forms.ControlStyles** bit, use the **System.Windows.Forms.Control.GetStyle(System.Windows.Forms.ControlStyles)** method and pass in the **System.Windows.Forms.ControlStyles** member to evaluate. The **System.Windows.Forms.ControlStyles** bit to set. **true** to apply the specified style to the control; otherwise, **false**.

```
rrrrrrrrrrrrrrrrrrrr)SetTopLevel
```

|           |            |          |                   |                   |
|-----------|------------|----------|-------------------|-------------------|
| [C#]      | protected  | void     | SetTopLevel(bool  | value);           |
| [C++]     | protected: | void     | SetTopLevel(bool  | value);           |
| [VB]      | Protected  | Sub      | SetTopLevel(ByVal | value As Boolean) |
| [JScript] | protected  | function | SetTopLevel(value | : Boolean);       |

### Description

Sets the control as the top-level control. **true** to set the control as the top-level control; otherwise, **false**.

```

ssssssssssssssssssssss)SetVisibleCore

```

|           |             |           |      |                                        |
|-----------|-------------|-----------|------|----------------------------------------|
| [C#]      | protected   | virtual   | void | SetVisibleCore(bool value);            |
| [C++]     | protected:  | virtual   | void | SetVisibleCore(bool value);            |
| [VB]      | Overridable | Protected | Sub  | SetVisibleCore(ByVal value As Boolean) |
| [JScript] | protected   | function  |      | SetVisibleCore(value : Boolean);       |

## Description

Sets the control to the specified visible state. **true** to make the control visible; otherwise, **false**.

### tttttttttt)Show

|           |         |          |         |
|-----------|---------|----------|---------|
| [C#]      | public  | void     | Show(); |
| [C++]     | public: | void     | Show(); |
| [VB]      | Public  | Sub      | Show()  |
| [JScript] | public  | function | Show(); |

## Description

Displays the control to the user.

Showing the control is equal to setting the **System.Windows.Forms.Control.Visible** property to **true**. After the **System.Windows.Forms.Control.Show** method is called the **System.Windows.Forms.Control.Visible** property returns a value of **true** until the **System.Windows.Forms.Control.Hide** method is called.

### uuuuuuuuuuuuuuuuuuuu)SuspendLayout

|           |         |          |                  |
|-----------|---------|----------|------------------|
| [C#]      | public  | void     | SuspendLayout(); |
| [C++]     | public: | void     | SuspendLayout(); |
| [VB]      | Public  | Sub      | SuspendLayout()  |
| [JScript] | public  | function | SuspendLayout(); |

## Description

Temporarily suspends the layout logic for the control.



xxxxxxxxxxxxxx)UnsafeNativeMethods.IOleControl.OnAmbientPropertyChange  
nge

[C#] int UnsafeNativeMethods.IOleControl.OnAmbientPropertyChange(int  
dispID);

[C++] int UnsafeNativeMethods::IOleControl::OnAmbientPropertyChange(int  
dispID);

[VB] Function IOleControl.OnAmbientPropertyChange(ByVal dispID As Integer)  
As Integer Implements

UnsafeNativeMethods.IOleControl.OnAmbientPropertyChange

[JScript] function

UnsafeNativeMethods.IOleControl.OnAmbientPropertyChange(dispID : int) : int;

yyyyyyyyyyyyyyyyyy)UnsafeNativeMethods.IOleControl.OnMnemonic

[C#] int UnsafeNativeMethods.IOleControl.OnMnemonic(ref  
NativeMethods.MSG pMsg);

[C++] int  
UnsafeNativeMethods::IOleControl::OnMnemonic(NativeMethods.MSG\* pMsg);

[VB] Function IOleControl.OnMnemonic(ByRef pMsg As NativeMethods.MSG)  
As Integer Implements UnsafeNativeMethods.IOleControl.OnMnemonic

[JScript] function UnsafeNativeMethods.IOleControl.OnMnemonic(pMsg :  
NativeMethods.MSG) : int;

zzzzzzzzzzzzzzzzzzzz)UnsafeNativeMethods.IOleInPlaceActiveObject.ContextSensitive  
Help

[C#] void

```

1 UnsafeNativeMethods.IOleInPlaceActiveObject.ContextSensitiveHelp(int
2 fEnterMode);
3 [C++] void
4 UnsafeNativeMethods::IOleInPlaceActiveObject::ContextSensitiveHelp(int
5 fEnterMode);
6 [VB] Sub IOleInPlaceActiveObject.ContextSensitiveHelp(ByVal fEnterMode As
7 Integer) Implements
8 UnsafeNativeMethods.IOleInPlaceActiveObject.ContextSensitiveHelp
9 [JScript] function
10 UnsafeNativeMethods.IOleInPlaceActiveObject.ContextSensitiveHelp(fEnterMod
11 e : int);
12 aaaaaaaaaaaaaaaa)UnsafeNativeMethods.IOleInPlaceActiveObject.EnableMo
13 deless
14
15 [C#] void UnsafeNativeMethods.IOleInPlaceActiveObject.EnableModeless(int
16 fEnable);
17 [C++] void UnsafeNativeMethods::IOleInPlaceActiveObject::EnableModeless(int
18 fEnable);
19 [VB] Sub IOleInPlaceActiveObject.EnableModeless(ByVal fEnable As Integer)
20 Implements UnsafeNativeMethods.IOleInPlaceActiveObject.EnableModeless
21 [JScript] function
22 UnsafeNativeMethods.IOleInPlaceActiveObject.EnableModeless(fEnable : int);
23
24
25

```

**bbbbbbbbbbbbbbbb)UnsafeNativeMethods.IoleInPlaceActiveObject.GetWindow**

**w**

[C#] int UnsafeNativeMethods.IoleInPlaceActiveObject.GetWindow(out IntPtr  
hwnd);

[C++] int UnsafeNativeMethods::IoleInPlaceActiveObject::GetWindow(IntPtr\*  
hwnd);

[VB] Function IoleInPlaceActiveObject.GetWindow(ByRef hwnd As IntPtr) As  
Integer Implements UnsafeNativeMethods.IoleInPlaceActiveObject.GetWindow

[JScript] function

UnsafeNativeMethods.IoleInPlaceActiveObject.GetWindow(hwnd : IntPtr) : int;

**cccccccccccccccc)UnsafeNativeMethods.IoleInPlaceActiveObject.OnDocWindowActivate**

[C#] void  
UnsafeNativeMethods.IoleInPlaceActiveObject.OnDocWindowActivate(int  
fActivate);

[C++] void  
UnsafeNativeMethods::IoleInPlaceActiveObject::OnDocWindowActivate(int  
fActivate);

[VB] Sub IoleInPlaceActiveObject.OnDocWindowActivate(ByVal fActivate As  
Integer) Implements

UnsafeNativeMethods.IoleInPlaceActiveObject.OnDocWindowActivate

[JScript] function

UnsafeNativeMethods.IoleInPlaceActiveObject.OnDocWindowActivate(fActivate : int);

**dddddddddddddddd)UnsafeNativeMethods.IoleInPlaceActiveObject.OnFrame  
WindowActivate**

[C#] void  
UnsafeNativeMethods.IoleInPlaceActiveObject.OnFrameWindowActivate(int  
fActivate);

[C++] void  
UnsafeNativeMethods::IoleInPlaceActiveObject::OnFrameWindowActivate(int  
fActivate);

[VB] Sub IoleInPlaceActiveObject.OnFrameWindowActivate(ByVal fActivate  
As Integer) Implements

UnsafeNativeMethods.IoleInPlaceActiveObject.OnFrameWindowActivate

[JScript] function  
UnsafeNativeMethods.IoleInPlaceActiveObject.OnFrameWindowActivate(fActiv  
ate : int);

**eeeeeeeeeeeeeeee)UnsafeNativeMethods.IoleInPlaceActiveObject.ResizeBorder**

[C#] void  
UnsafeNativeMethods.IoleInPlaceActiveObject.ResizeBorder(NativeMethods.CO  
MRECT prcBorder, UnsafeNativeMethods.IoleInPlaceUIWindow pUIWindow,  
int fFrameWindow);

[C++] void  
UnsafeNativeMethods::IoleInPlaceActiveObject::ResizeBorder(NativeMethods.C  
OMRECT\* prcBorder, UnsafeNativeMethods.IoleInPlaceUIWindow\*  
pUIWindow, int fFrameWindow);

```

1 [VB] Sub IOleInPlaceActiveObject.ResizeBorder(ByVal prcBorder As
2 NativeMethods.COMRECT, ByVal pUIWindow As
3 UnsafeNativeMethods.IOleInPlaceUIWindow, ByVal fFrameWindow As Integer)
4 Implements UnsafeNativeMethods.IOleInPlaceActiveObject.ResizeBorder
5 [JScript] function
6 UnsafeNativeMethods.IOleInPlaceActiveObject.ResizeBorder(prcBorder :
7 NativeMethods.COMRECT, pUIWindow :
8 UnsafeNativeMethods.IOleInPlaceUIWindow, fFrameWindow : int);
9 ffffffffffffffff)UnsafeNativeMethods.IOleInPlaceActiveObject.TranslateAccelerator
10
11 [C#] int UnsafeNativeMethods.IOleInPlaceActiveObject.TranslateAccelerator(ref
12 NativeMethods.MSG lpmsg);
13 [C++] int
14 UnsafeNativeMethods::IOleInPlaceActiveObject::TranslateAccelerator(NativeMet
15 hods.MSG* lpmsg);
16 [VB] Function IOleInPlaceActiveObject.TranslateAccelerator(ByRef lpmsg As
17 NativeMethods.MSG) As Integer Implements
18 UnsafeNativeMethods.IOleInPlaceActiveObject.TranslateAccelerator
19 [JScript] function
20 UnsafeNativeMethods.IOleInPlaceActiveObject.TranslateAccelerator(lpmsg :
21 NativeMethods.MSG) : int;
22
23
24
25

```



**gggggggggggggggggg)UnsafeNativeMethods.IoleInPlaceObject.ContextSensitive  
Help**

[C#] void UnsafeNativeMethods.IoleInPlaceObject.ContextSensitiveHelp(int  
fEnterMode);

[C++] void UnsafeNativeMethods::IoleInPlaceObject::ContextSensitiveHelp(int  
fEnterMode);

[VB] Sub IoleInPlaceObject.ContextSensitiveHelp(ByVal fEnterMode As  
Integer) Implements

UnsafeNativeMethods.IoleInPlaceObject.ContextSensitiveHelp

[JScript] function

UnsafeNativeMethods.IoleInPlaceObject.ContextSensitiveHelp(fEnterMode :  
int);

**hhhhhhhhhhhhhhhh)UnsafeNativeMethods.IoleInPlaceObject.GetWindow**

[C#] int UnsafeNativeMethods.IoleInPlaceObject.GetWindow(out IntPtr hwnd);

[C++] int UnsafeNativeMethods::IoleInPlaceObject::GetWindow(IntPtr\* hwnd);

[VB] Function IoleInPlaceObject.GetWindow(ByRef hwnd As IntPtr) As Integer  
Implements UnsafeNativeMethods.IoleInPlaceObject.GetWindow

[JScript] function UnsafeNativeMethods.IoleInPlaceObject.GetWindow(hwnd :  
IntPtr) : int;

**iiiiiiiiiiiiiii)UnsafeNativeMethods.IoleInPlaceObject.InPlaceDeactivate**

[C#] void UnsafeNativeMethods.IoleInPlaceObject.InPlaceDeactivate();

[C++] void UnsafeNativeMethods::IoleInPlaceObject::InPlaceDeactivate();

```

1  [VB]      Sub      IOleInPlaceObject.InPlaceDeactivate()      Implements
2  UnsafeNativeMethods.IOleInPlaceObject.InPlaceDeactivate
3  [JScript] function UnsafeNativeMethods.IOleInPlaceObject.InPlaceDeactivate();
4
5
6  [C#]      void      UnsafeNativeMethods.IOleInPlaceObject.ReactivateAndUndo();
7  [C++]      void      UnsafeNativeMethods::IOleInPlaceObject::ReactivateAndUndo();
8  [VB]      Sub      IOleInPlaceObject.ReactivateAndUndo()      Implements
9  UnsafeNativeMethods.IOleInPlaceObject.ReactivateAndUndo
10 [JScript]
11
12
13
14 [C#]
15
16
17 [C++]
18
19
20 [VB]      Sub      IOleInPlaceObject.SetObjectRects(ByVal lprcPosRect As
21 NativeMethods.COMRECT, ByVal lprcClipRect As NativeMethods.COMRECT)
22 Implements      UnsafeNativeMethods.IOleInPlaceObject.SetObjectRects
23 [JScript]
24
25

```

```

1 UnsafeNativeMethods.IOleInPlaceObject.SetObjectRects(lprcPosRect      :
2 NativeMethods.COMRECT, lprcClipRect : NativeMethods.COMRECT);

```

***UUUUUUUUUUUU)UnsafeNativeMethods.IOleInPlaceObject.UIDeactivate***

```

5 [C#]      int      UnsafeNativeMethods.IOleInPlaceObject.UIDeactivate();
6 [C++]      int      UnsafeNativeMethods::IOleInPlaceObject::UIDeactivate();
7 [VB]  Function  IOleInPlaceObject.UIDeactivate()  As  Integer  Implements
8 UnsafeNativeMethods.IOleInPlaceObject.UIDeactivate
9 [JScript] function UnsafeNativeMethods.IOleInPlaceObject.UIDeactivate() : int;

```

***MMMMMMMMMMMMMMMMMMMM)UnsafeNativeMethods.IOleObject.Advise***

```

12 [C#]                                             int
13 UnsafeNativeMethods.IOleObject.Advise(UnsafeNativeMethods.IAdviseSink
14 pAdvSink,                                     out      int      cookie);
15 [C++]                                             int
16 UnsafeNativeMethods::IOleObject::Advise(UnsafeNativeMethods.IAdviseSink*
17 pAdvSink,                                     int*      cookie);
18 [VB]      Function      IOleObject.Advise(ByVal      pAdvSink      As
19 UnsafeNativeMethods.IAdviseSink, ByRef cookie As Integer) As Integer
20 Implements                                     UnsafeNativeMethods.IOleObject.Advise
21 [JScript] function      UnsafeNativeMethods.IOleObject.Advise(pAdvSink      :
22 UnsafeNativeMethods.IAdviseSink, cookie : int) : int;

```



***pppppppppppppppppppppp)UnsafeNativeMethods.IOleObject.EnumAdvise***

```
1
2
3 [C#]          int          UnsafeNativeMethods.IOleObject.EnumAdvise(out
4 UnsafeNativeMethods.IEnumSTATDATA                                     e);
5
6 [C++]                                     int
7 UnsafeNativeMethods::IOleObject::EnumAdvise(UnsafeNativeMethods.IEnumS
8 TATDATA**                                     e);
9
10 [VB]          Function      IOleObject.EnumAdvise(ByRef      e      As
11 UnsafeNativeMethods.IEnumSTATDATA)      As      Integer      Implements
12 UnsafeNativeMethods.IOleObject.EnumAdvise
13
14 [JScript]      function      UnsafeNativeMethods.IOleObject.EnumAdvise(e      :
15 UnsafeNativeMethods.IEnumSTATDATA) : int;
```

***qqqqqqqqqqqqqqqqqqqq)UnsafeNativeMethods.IOleObject.EnumVerbs***

```
14
15 [C#]          int          UnsafeNativeMethods.IOleObject.EnumVerbs(out
16 UnsafeNativeMethods.IEnumOLEVERB                                     e);
17
18 [C++]                                     int
19 UnsafeNativeMethods::IOleObject::EnumVerbs(UnsafeNativeMethods.IEnumOL
20 EVERB**                                     e);
21
22 [VB]          Function      IOleObject.EnumVerbs(ByRef      e      As
23 UnsafeNativeMethods.IEnumOLEVERB)      As      Integer      Implements
24 UnsafeNativeMethods.IOleObject.EnumVerbs
25
26 [JScript]      function      UnsafeNativeMethods.IOleObject.EnumVerbs(e      :
27 UnsafeNativeMethods.IEnumOLEVERB) : int;
```



***tttttttttttt)UnsafeNativeMethods.IOleObject.GetExtent***

[C#] int UnsafeNativeMethods.IOleObject.GetExtent(int dwDrawAspect,  
NativeMethods.tagSIZEL pSizel);

[C++] int UnsafeNativeMethods::IOleObject::GetExtent(int dwDrawAspect,  
NativeMethods.tagSIZEL\* pSizel);

[VB] Function IOleObject.GetExtent(ByVal dwDrawAspect As Integer, ByVal  
pSizel As NativeMethods.tagSIZEL) As Integer Implements  
UnsafeNativeMethods.IOleObject.GetExtent

[JScript] function UnsafeNativeMethods.IOleObject.GetExtent(dwDrawAspect :  
int, pSizel : NativeMethods.tagSIZEL) : int;

***uuuuuuuuuuuuuuuuuu)UnsafeNativeMethods.IOleObject.GetMiscStatus***

[C#] int UnsafeNativeMethods.IOleObject.GetMiscStatus(int dwAspect, out int  
cookie);

[C++] int UnsafeNativeMethods::IOleObject::GetMiscStatus(int dwAspect, int\*  
cookie);

[VB] Function IOleObject.GetMiscStatus(ByVal dwAspect As Integer, ByRef  
cookie As Integer) As Integer Implements  
UnsafeNativeMethods.IOleObject.GetMiscStatus

[JScript] function UnsafeNativeMethods.IOleObject.GetMiscStatus(dwAspect :  
int, cookie : int) : int;

***vvvvvvvvvvvvvvvvvv)UnsafeNativeMethods.IOleObject.GetMoniker***

[C#] int UnsafeNativeMethods.IOleObject.GetMoniker(int dwAssign, int





```

1 UnsafeNativeMethods.IOleObject.GetUserType
2 [JScript] function
3 UnsafeNativeMethods.IOleObject.GetUserType(dwFormOfType : int, userType :
4 String) : int;
5 yyyyyyyyyyyyyyyyyy)UnsafeNativeMethods.IOleObject.InitFromData
6
7 [C#] int
8 UnsafeNativeMethods.IOleObject.InitFromData(UnsafeNativeMethods.IOleData
9 Object pDataObject, int fCreation, int dwReserved);
10 [C++] int
11 UnsafeNativeMethods::IOleObject::InitFromData(UnsafeNativeMethods.IOleDat
12 aObject* pDataObject, int fCreation, int dwReserved);
13 [VB] Function IOleObject.InitFromData(ByVal pDataObject As
14 UnsafeNativeMethods.IOleDataObject, ByVal fCreation As Integer, ByVal
15 dwReserved As Integer) As Integer Implements
16 UnsafeNativeMethods.IOleObject.InitFromData
17 [JScript] function UnsafeNativeMethods.IOleObject.InitFromData(pDataObject :
18 UnsafeNativeMethods.IOleDataObject, fCreation : int, dwReserved : int) : int;
19 zzzzzzzzzzzzzzzzzzzz)UnsafeNativeMethods.IOleObject.IsUpToDate
20
21 [C#] int UnsafeNativeMethods.IOleObject.IsUpToDate();
22 [C++] int UnsafeNativeMethods::IOleObject::IsUpToDate();
23 [VB] Function IOleObject.IsUpToDate() As Integer Implements
24
25

```

```

1 UnsafeNativeMethods.IOleObject.IsUpToDate
2 [JScript] function UnsafeNativeMethods.IOleObject.IsUpToDate() : int;
3 aaaaaaaaaaaaaaaaaaaa)UnsafeNativeMethods.IOleObject.OleUpdate
4
5 [C#] int UnsafeNativeMethods.IOleObject.OleUpdate();
6 [C++] int UnsafeNativeMethods::IOleObject::OleUpdate();
7 [VB] Function IOleObject.OleUpdate() As Integer Implements
8 UnsafeNativeMethods.IOleObject.OleUpdate
9 [JScript] function UnsafeNativeMethods.IOleObject.OleUpdate() : int;
10 bbbbbbbbbbbbbbbbbbbb)UnsafeNativeMethods.IOleObject.SetClientSite
11
12 [C#] int
13 UnsafeNativeMethods.IOleObject.SetClientSite(UnsafeNativeMethods.IOleClient
14 Site pClientSite);
15 [C++] int
16 UnsafeNativeMethods::IOleObject::SetClientSite(UnsafeNativeMethods.IOleClie
17 ntSite* pClientSite);
18 [VB] Function IOleObject.SetClientSite(ByVal pClientSite As
19 UnsafeNativeMethods.IOleClientSite) As Integer Implements
20 UnsafeNativeMethods.IOleObject.SetClientSite
21 [JScript] function UnsafeNativeMethods.IOleObject.SetClientSite(pClientSite :
22 UnsafeNativeMethods.IOleClientSite) : int;
23
24
25

```

*cccccccccccccccc)UnsafeNativeMethods.IOleObject.SetColorScheme*

```
1
2
3 [C#] int
4 UnsafeNativeMethods.IOleObject.SetColorScheme(NativeMethods.tagLOGPALE
5 TTE pLogpal);
6 [C++] int
7 UnsafeNativeMethods::IOleObject::SetColorScheme(NativeMethods.tagLOGPAL
8 ETTE* pLogpal);
9 [VB] Function IOleObject.SetColorScheme(ByVal pLogpal As
10 NativeMethods.tagLOGPALETTE) As Integer Implements
11 UnsafeNativeMethods.IOleObject.SetColorScheme
12 [JScript] function UnsafeNativeMethods.IOleObject.SetColorScheme(pLogpal :
13 NativeMethods.tagLOGPALETTE) : int;
```

*dddddddddddddddd)UnsafeNativeMethods.IOleObject.SetExtent*

```
14
15
16 [C#] int UnsafeNativeMethods.IOleObject.SetExtent(int dwDrawAspect,
17 NativeMethods.tagSIZEL pSizel);
18 [C++] int UnsafeNativeMethods::IOleObject::SetExtent(int dwDrawAspect,
19 NativeMethods.tagSIZEL* pSizel);
20 [VB] Function IOleObject.SetExtent(ByVal dwDrawAspect As Integer, ByVal
21 pSizel As NativeMethods.tagSIZEL) As Integer Implements
22 UnsafeNativeMethods.IOleObject.SetExtent
23 [JScript] function UnsafeNativeMethods.IOleObject.SetExtent(dwDrawAspect :
24 int, pSizel : NativeMethods.tagSIZEL) : int;
```

*eeeeeeeeeeeeeeee)UnsafeNativeMethods.IOleObject.SetHostNames*

[C#] int UnsafeNativeMethods.IOleObject.SetHostNames(string szContainerApp,  
string szContainerObj);

[C++] int UnsafeNativeMethods::IOleObject::SetHostNames(String\*  
szContainerApp, String\* szContainerObj);

[VB] Function IOleObject.SetHostNames(ByVal szContainerApp As String,  
ByVal szContainerObj As String) As Integer Implements  
UnsafeNativeMethods.IOleObject.SetHostNames

[JScript] function  
UnsafeNativeMethods.IOleObject.SetHostNames(szContainerApp : String,  
szContainerObj : String) : int;

*ffffffffffffffff)UnsafeNativeMethods.IOleObject.SetMoniker*

[C#] int UnsafeNativeMethods.IOleObject.SetMoniker(int dwWhichMoniker,  
object pmk);

[C++] int UnsafeNativeMethods::IOleObject::SetMoniker(int dwWhichMoniker,  
Object\* pmk);

[VB] Function IOleObject.SetMoniker(ByVal dwWhichMoniker As Integer,  
ByVal pmk As Object) As Integer Implements  
UnsafeNativeMethods.IOleObject.SetMoniker

[JScript] function  
UnsafeNativeMethods.IOleObject.SetMoniker(dwWhichMoniker : int, pmk :  
Object) : int;

**gggggggggggggggggggg)UnsafeNativeMethods.IOleObject.Unadvise**

[C#] int UnsafeNativeMethods.IOleObject.Unadvise(int dwConnection);  
[C++] int UnsafeNativeMethods::IOleObject::Unadvise(int dwConnection);  
[VB] Function IOleObject.Unadvise(ByVal dwConnection As Integer) As Integer  
Implements UnsafeNativeMethods.IOleObject.Unadvise  
[JScript] function UnsafeNativeMethods.IOleObject.Unadvise(dwConnection :  
int) : int;

**hhhhhhhhhhhhhhhhhh)UnsafeNativeMethods.IOleWindow.ContextSensitiveHelp  
lp**

[C#] void UnsafeNativeMethods.IOleWindow.ContextSensitiveHelp(int  
fEnterMode);  
[C++] void UnsafeNativeMethods::IOleWindow::ContextSensitiveHelp(int  
fEnterMode);  
[VB] Sub IOleWindow.ContextSensitiveHelp(ByVal fEnterMode As Integer)  
Implements UnsafeNativeMethods.IOleWindow.ContextSensitiveHelp  
[JScript] function  
UnsafeNativeMethods.IOleWindow.ContextSensitiveHelp(fEnterMode : int);

**iiiiiiiiiiiiiii)UnsafeNativeMethods.IOleWindow.GetWindow**

[C#] int UnsafeNativeMethods.IOleWindow.GetWindow(out IntPtr hwnd);  
[C++] int UnsafeNativeMethods::IOleWindow::GetWindow(IntPtr\* hwnd);  
[VB] Function IOleWindow.GetWindow(ByRef hwnd As IntPtr) As Integer  
Implements UnsafeNativeMethods.IOleWindow.GetWindow

1 [JScript] function UnsafeNativeMethods.IOleWindow.GetWindow(hwnd : IntPtr)  
2 : int;

3 *))))))))))UnsafeNativeMethods.IPersist.GetClassID*

4  
5 [C#] void UnsafeNativeMethods.IPersist.GetClassID(out Guid pClassID);

6 [C++] void UnsafeNativeMethods::IPersist::GetClassID(Guid\* pClassID);

7 [VB] Sub IPersist.GetClassID(ByRef pClassID As Guid) Implements

8 UnsafeNativeMethods.IPersist.GetClassID

9 [JScript] function UnsafeNativeMethods.IPersist.GetClassID(pClassID : Guid);

10 *))))))))))))))UnsafeNativeMethods.IPersistPropertyBag.GetClassID*

11  
12 [C#] void UnsafeNativeMethods.IPersistPropertyBag.GetClassID(out Guid

13 pClassID);

14 [C++] void UnsafeNativeMethods::IPersistPropertyBag::GetClassID(Guid\*

15 pClassID);

16 [VB] Sub IPersistPropertyBag.GetClassID(ByRef pClassID As Guid) Implements

17 UnsafeNativeMethods.IPersistPropertyBag.GetClassID

18 [JScript] function

19 UnsafeNativeMethods.IPersistPropertyBag.GetClassID(pClassID : Guid);

20 *))))))))))))UnsafeNativeMethods.IPersistPropertyBag.InitNew*

21  
22 [C#] void UnsafeNativeMethods.IPersistPropertyBag.InitNew();

23 [C++] void UnsafeNativeMethods::IPersistPropertyBag::InitNew();

24 [VB] Sub IPersistPropertyBag.InitNew() Implements

[illegible]

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
84

mmmmmmmmmmmmmmmmmmmmmmmm)UnsafeNativeMethods.IPersistPropertyBag.Lo  
ad

[C#] void

```
UnsafeNativeMethods.IPersistPropertyBag.Load(UnsafeNativeMethods.IProperty
Bag pPropBag, UnsafeNativeMethods.IErrorLog pErrorLog);
```

```
[C++] void
```

```
UnsafeNativeMethods::IPersistPropertyBag::Load(UnsafeNativeMethods.IPropert
yBag* pPropBag, UnsafeNativeMethods.IErrorLog* pErrorLog);
```

|      |     |                                   |          |            |
|------|-----|-----------------------------------|----------|------------|
| [VB] | Sub | IPersistPropertyBag.Load(ByVal    | pPropBag | As         |
|      |     | UnsafeNativeMethods.IPropertyBag, | ByVal    | pErrorLog  |
|      |     | UnsafeNativeMethods.IErrorLog)    |          | Implements |

## UnsafeNativeMethods.IPersistPropertyBag.Load

```
[JScript] function UnsafeNativeMethods.IPersistPropertyBag.Load(pPropBag :
UnsafeNativeMethods.IPropertyBag, pErrorLog :
UnsafeNativeMethods.IErrorLog);
```

```
nnnnnnnnnnnnnnnnnnnn)UnsafeNativeMethods.IPersistPropertyBag.Save
```

|      |      |
|------|------|
| [C#] | void |
|------|------|

```
UnsafeNativeMethods.IPersistPropertyBag.Save(UnsafeNativeMethods.IProperty
Bag    pPropBag,    bool    fClearDirty,    bool    fSaveAllProperties);
```

```

[C++] void

```

```
UnsafeNativeMethods::IPersistPropertyBag::Save(UnsafeNativeMethods.IPropert
```

```

1 yBag* pPropBag, bool fClearDirty, bool fSaveAllProperties);
2 [VB] Sub IPersistPropertyBag.Save(ByVal pPropBag As
3 UnsafeNativeMethods.IPropertyBag, ByVal fClearDirty As Boolean, ByVal
4 fSaveAllProperties As Boolean) Implements
5 UnsafeNativeMethods.IPersistPropertyBag.Save
6 [JScript] function UnsafeNativeMethods.IPersistPropertyBag.Save(pPropBag :
7 UnsafeNativeMethods.IPropertyBag, fClearDirty : Boolean, fSaveAllProperties :
8 Boolean);
9 oooooooooooooooooooo)UnsafeNativeMethods.IPersistStorage.GetClassID
10
11 [C#] void UnsafeNativeMethods.IPersistStorage.GetClassID(out Guid pClassID);
12 [C++] void UnsafeNativeMethods::IPersistStorage::GetClassID(Guid* pClassID);
13 [VB] Sub IPersistStorage.GetClassID(ByRef pClassID As Guid) Implements
14 UnsafeNativeMethods.IPersistStorage.GetClassID
15 [JScript] function UnsafeNativeMethods.IPersistStorage.GetClassID(pClassID :
16 Guid);
17 pppppppppppppppppppp)UnsafeNativeMethods.IPersistStorage.HandsOffStorage
18
19 [C#] void UnsafeNativeMethods.IPersistStorage.HandsOffStorage();
20 [C++] void UnsafeNativeMethods::IPersistStorage::HandsOffStorage();
21 [VB] Sub IPersistStorage.HandsOffStorage() Implements
22 UnsafeNativeMethods.IPersistStorage.HandsOffStorage
23 [JScript] function UnsafeNativeMethods.IPersistStorage.HandsOffStorage();
24
25

```



**qqqqqqqqqqqqqqqqqq)UnsafeNativeMethods.IPersistStorage.InitNew**

[C#] void

UnsafeNativeMethods.IPersistStorage.InitNew(UnsafeNativeMethods.IStorage  
pstg);

[C++] void

UnsafeNativeMethods::IPersistStorage::InitNew(UnsafeNativeMethods.IStorage\*  
pstg);

[VB] Sub IPersistStorage.InitNew(ByVal pstg As UnsafeNativeMethods.IStorage)

Implements UnsafeNativeMethods.IPersistStorage.InitNew

[JScript] function UnsafeNativeMethods.IPersistStorage.InitNew(pstg :  
UnsafeNativeMethods.IStorage);

**rrrrrrrrrrrrrrrrrr)UnsafeNativeMethods.IPersistStorage.IsDirty**

[C#] int UnsafeNativeMethods.IPersistStorage.IsDirty();

[C++] int UnsafeNativeMethods::IPersistStorage::IsDirty();

[VB] Function IPersistStorage.IsDirty() As Integer Implements  
UnsafeNativeMethods.IPersistStorage.IsDirty

[JScript] function UnsafeNativeMethods.IPersistStorage.IsDirty() : int;

**ssssssssssssssss)UnsafeNativeMethods.IPersistStorage.Load**

[C#] int

UnsafeNativeMethods.IPersistStorage.Load(UnsafeNativeMethods.IStorage pstg);

[C++] int

UnsafeNativeMethods::IPersistStorage::Load(UnsafeNativeMethods.IStorage\*

```

1  pstg);
2  [VB]      Function      IPersistStorage.Load(ByVal      pstg      As
3  UnsafeNativeMethods.IStorage)      As      Integer      Implements
4  UnsafeNativeMethods.IPersistStorage.Load
5  [JScript]      function      UnsafeNativeMethods.IPersistStorage.Load(pstg      :
6  UnsafeNativeMethods.IStorage) : int;
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

```

**VBScript:**  
**Function** IPersistStorage.Load(**ByVal** pstg As UnsafeNativeMethods.IStorage) As Integer Implements UnsafeNativeMethods.IPersistStorage.Load  
 [JScript] function UnsafeNativeMethods.IPersistStorage.Load(pstg : UnsafeNativeMethods.IStorage) : int;  
*#####UnsafeNativeMethods.IPersistStorage.Save*

**C#:**  
 void  
 UnsafeNativeMethods.IPersistStorage.Save(UnsafeNativeMethods.IStorage pstg, int fSameAsLoad);

**C++:**  
 void  
 UnsafeNativeMethods::IPersistStorage::Save(UnsafeNativeMethods.IStorage\* pstg, int fSameAsLoad);

**VBScript:**  
**Sub** IPersistStorage.Save(**ByVal** pstg As UnsafeNativeMethods.IStorage, **ByVal** fSameAsLoad As Integer) Implements UnsafeNativeMethods.IPersistStorage.Save  
 [JScript] function UnsafeNativeMethods.IPersistStorage.Save(pstg : UnsafeNativeMethods.IStorage, fSameAsLoad : int);  
*#####UnsafeNativeMethods.IPersistStorage.SaveCompleted*

**C#:**  
 void  
 UnsafeNativeMethods.IPersistStorage.SaveCompleted(UnsafeNativeMethods.IStorage pstgNew);



```

1 UnsafeNativeMethods.IPersistStreamInit.GetSizeMax
2 [JScript] function UnsafeNativeMethods.IPersistStreamInit.GetSizeMax(pcbSize :
3 long);
4 xxxxxxxxxxxxxxxxxx)UnsafeNativeMethods.IPersistStreamInit.InitNew
5
6 [C#] void UnsafeNativeMethods.IPersistStreamInit.InitNew();
7 [C++] void UnsafeNativeMethods::IPersistStreamInit::InitNew();
8 [VB] Sub IPersistStreamInit.InitNew() Implements
9 UnsafeNativeMethods.IPersistStreamInit.InitNew
10 [JScript] function UnsafeNativeMethods.IPersistStreamInit.InitNew();
11 yyyyyyyyyyyyyyyyyy)UnsafeNativeMethods.IPersistStreamInit.IsDirty
12
13 [C#] int UnsafeNativeMethods.IPersistStreamInit.IsDirty();
14 [C++] int UnsafeNativeMethods::IPersistStreamInit::IsDirty();
15 [VB] Function IPersistStreamInit.IsDirty() As Integer Implements
16 UnsafeNativeMethods.IPersistStreamInit.IsDirty
17 [JScript] function UnsafeNativeMethods.IPersistStreamInit.IsDirty() : int;
18 zzzzzzzzzzzzzzzzzz)UnsafeNativeMethods.IPersistStreamInit.Load
19
20 [C#] void
21 UnsafeNativeMethods.IPersistStreamInit.Load(UnsafeNativeMethods.IStream
22 pstm);
23 [C++] void
24 UnsafeNativeMethods::IPersistStreamInit::Load(UnsafeNativeMethods.IStream*
25

```

```

1  pstm);
2  [VB] Sub IPersistStreamInit.Load(ByVal pstm As UnsafeNativeMethods.IStream)
3  Implements                                     UnsafeNativeMethods.IPersistStreamInit.Load
4  [JScript]   function   UnsafeNativeMethods.IPersistStreamInit.Load(pstm   :
5  UnsafeNativeMethods.IStream);
6
7
8  [C#]                                               void
9  UnsafeNativeMethods.IPersistStreamInit.Save(UnsafeNativeMethods.IStream
10 pstm,                                               bool                               fClearDirty);
11 [C++]                                               void
12 UnsafeNativeMethods::IPersistStreamInit::Save(UnsafeNativeMethods.IStream*
13 pstm,                                               bool                               fClearDirty);
14 [VB] Sub IPersistStreamInit.Save(ByVal pstm As UnsafeNativeMethods.IStream,
15 ByVal fClearDirty As Boolean) Implements
16 UnsafeNativeMethods.IPersistStreamInit.Save
17 [JScript]   function   UnsafeNativeMethods.IPersistStreamInit.Save(pstm   :
18 UnsafeNativeMethods.IStream, fClearDirty : Boolean);
19
20
21 [C#]                                               void
22 UnsafeNativeMethods.IQuickActivate.GetContentExtent(NativeMethods.tagSIZE
23 L                                               pSizel);
24 [C++]                                               void
25

```

```

1 UnsafeNativeMethods::IQuickActivate::GetContentExtent(NativeMethods.tagSIZ
2 EL* pSizel);
3 [VB] Sub IQuickActivate.GetContentExtent(ByVal pSizel As
4 NativeMethods.tagSIZEL) Implements
5 UnsafeNativeMethods.IQuickActivate.GetContentExtent
6 [JScript] function UnsafeNativeMethods.IQuickActivate.GetContentExtent(pSizel
7 : NativeMethods.tagSIZEL);
8 cccccccccccccccc)UnsafeNativeMethods.IQuickActivate.QuickActivate
9
10 [C#] void
11 UnsafeNativeMethods.IQuickActivate.QuickActivate(UnsafeNativeMethods.tagQ
12 ACONTAINER pQaContainer, UnsafeNativeMethods.tagQACONTROL
13 pQaControl);
14 [C++] void
15 UnsafeNativeMethods::IQuickActivate::QuickActivate(UnsafeNativeMethods.tag
16 QACONTAINER* pQaContainer, UnsafeNativeMethods.tagQACONTROL*
17 pQaControl);
18 [VB] Sub IQuickActivate.QuickActivate(ByVal pQaContainer As
19 UnsafeNativeMethods.tagQACONTAINER, ByVal pQaControl As
20 UnsafeNativeMethods.tagQACONTROL) Implements
21 UnsafeNativeMethods.IQuickActivate.QuickActivate
22 [JScript] function
23 UnsafeNativeMethods.IQuickActivate.QuickActivate(pQaContainer :
24 UnsafeNativeMethods.tagQACONTAINER, pQaControl :
25 UnsafeNativeMethods.tagQACONTROL);

```

*dddddddddddddddd)UnsafeNativeMethods.IQuickActivate.SetContentExtent*

[C#] void

UnsafeNativeMethods.IQuickActivate.SetContentExtent(NativeMethods.tagSIZE  
L pSizel);

[C++] void

UnsafeNativeMethods::IQuickActivate::SetContentExtent(NativeMethods.tagSIZ  
EL\* pSizel);

[VB] Sub IQuickActivate.SetContentExtent(ByVal pSizel As  
NativeMethods.tagSIZEL) Implements

UnsafeNativeMethods.IQuickActivate.SetContentExtent

[JScript] function UnsafeNativeMethods.IQuickActivate.SetContentExtent(pSizel  
: NativeMethods.tagSIZEL);

*eeeeeeeeeeeeeeee)UnsafeNativeMethods.IViewObject.Draw*

[C#] void UnsafeNativeMethods.IViewObject.Draw(int dwDrawAspect, int  
lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE ptd, IntPtr  
hdcTargetDev, IntPtr hdcDraw, NativeMethods.COMRECT lprcBounds,  
NativeMethods.COMRECT lprcWBounds, IntPtr pfnContinue, int dwContinue);

[C++] void UnsafeNativeMethods::IViewObject::Draw(int dwDrawAspect, int  
lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE\* ptd, IntPtr  
hdcTargetDev, IntPtr hdcDraw, NativeMethods.COMRECT\* lprcBounds,  
NativeMethods.COMRECT\* lprcWBounds, IntPtr pfnContinue, int dwContinue);

[VB] Sub IViewObject.Draw(ByVal dwDrawAspect As Integer, ByVal lindex As  
Integer, ByVal pvAspect As IntPtr, ByVal ptd As

```

1 NativeMethods.tagDVTARGETDEVICE, ByVal hdcTargetDev As IntPtr, ByVal
2 hdcDraw As IntPtr, ByVal lprcBounds As NativeMethods.COMRECT, ByVal
3 lprcWBounds As NativeMethods.COMRECT, ByVal pfnContinue As IntPtr,
4 ByVal dwContinue As Integer) Implements
5 UnsafeNativeMethods.IViewObject.Draw
6 [JScript] function UnsafeNativeMethods.IViewObject.Draw(dwDrawAspect : int,
7 lindex : int, pvAspect : IntPtr, ptd : NativeMethods.tagDVTARGETDEVICE,
8 hdcTargetDev : IntPtr, hdcDraw : IntPtr, lprcBounds :
9 NativeMethods.COMRECT, lprcWBounds : NativeMethods.COMRECT,
10 pfnContinue : IntPtr, dwContinue : int);

```

***ffffffffffffffffUnsafeNativeMethods.IViewObject Freeze***

```

13 [C#] int UnsafeNativeMethods.IViewObject.Freeze(int dwDrawAspect, int lindex,
14 IntPtr pvAspect, IntPtr pdwFreeze);
15 [C++] int UnsafeNativeMethods::IViewObject::Freeze(int dwDrawAspect, int
16 lindex, IntPtr pvAspect, IntPtr pdwFreeze);
17 [VB] Function IViewObject.Freeze(ByVal dwDrawAspect As Integer, ByVal
18 lindex As Integer, ByVal pvAspect As IntPtr, ByVal pdwFreeze As IntPtr) As
19 Integer Implements UnsafeNativeMethods.IViewObject.Freeze
20 [JScript] function UnsafeNativeMethods.IViewObject.Freeze(dwDrawAspect :
21 int, lindex : int, pvAspect : IntPtr, pdwFreeze : IntPtr) : int;

```

***ggggggggggggggggggUnsafeNativeMethods.IViewObject.GetAdvise***

```

24 [C#] void UnsafeNativeMethods.IViewObject.GetAdvise(int[] paspects, int[]

```



```

1  padvf,          UnsafeNativeMethods.IAdviseSink[]          pAdvSink);
2  [C++] void UnsafeNativeMethods::IViewObject::GetAdvise(int paspects __gc[],
3  int padvf __gc[], UnsafeNativeMethods.IAdviseSink* pAdvSink[]);
4  [VB] Sub IViewObject.GetAdvise(ByVal paspects() As Integer, ByVal padvf() As
5  Integer, ByVal pAdvSink() As UnsafeNativeMethods.IAdviseSink) Implements
6  UnsafeNativeMethods.IViewObject.GetAdvise
7  [JScript] function UnsafeNativeMethods.IViewObject.GetAdvise(paspects : int[],
8  padvf : int[], pAdvSink : UnsafeNativeMethods.IAdviseSink[]);
9
10 hhhhhhhhhhhhhhhhhhhh)UnsafeNativeMethods.IViewObject.GetColorSet
11
12 [C#] int UnsafeNativeMethods.IViewObject.GetColorSet(int dwDrawAspect, int
13 lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE ptd, IntPtr
14 hicTargetDev, NativeMethods.tagLOGPALETTE ppColorSet);
15 [C++] int UnsafeNativeMethods::IViewObject::GetColorSet(int dwDrawAspect,
16 int lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE* ptd, IntPtr
17 hicTargetDev, NativeMethods.tagLOGPALETTE* ppColorSet);
18 [VB] Function IViewObject.GetColorSet(ByVal dwDrawAspect As Integer,
19 ByVal lindex As Integer, ByVal pvAspect As IntPtr, ByVal ptd As
20 NativeMethods.tagDVTARGETDEVICE, ByVal hicTargetDev As IntPtr, ByVal
21 ppColorSet As NativeMethods.tagLOGPALETTE) As Integer Implements
22 UnsafeNativeMethods.IViewObject.GetColorSet
23 [JScript]
24 function
25 UnsafeNativeMethods.IViewObject.GetColorSet(dwDrawAspect : int, lindex : int,
26 pvAspect : IntPtr, ptd : NativeMethods.tagDVTARGETDEVICE, hicTargetDev :
27 IntPtr, ppColorSet : NativeMethods.tagLOGPALETTE) : int;

```

**iiiiiiiiiiiiiiii)UnsafeNativeMethods.IViewObject.SetAdvise**

[C#] void UnsafeNativeMethods.IViewObject.SetAdvise(int aspects, int advf,  
UnsafeNativeMethods.IAdviseSink pAdvSink);

[C++] void UnsafeNativeMethods::IViewObject::SetAdvise(int aspects, int advf,  
UnsafeNativeMethods.IAdviseSink\* pAdvSink);

[VB] Sub IViewObject.SetAdvise(ByVal aspects As Integer, ByVal advf As  
Integer, ByVal pAdvSink As UnsafeNativeMethods.IAdviseSink) Implements  
UnsafeNativeMethods.IViewObject.SetAdvise

[JScript] function UnsafeNativeMethods.IViewObject.SetAdvise(aspects : int,  
advf : int, pAdvSink : UnsafeNativeMethods.IAdviseSink);

**jjjjjjjjjjjjjjjj)UnsafeNativeMethods.IViewObject.Unfreeze**

[C#] int UnsafeNativeMethods.IViewObject.Unfreeze(int dwFreeze);

[C++] int UnsafeNativeMethods::IViewObject::Unfreeze(int dwFreeze);

[VB] Function IViewObject.Unfreeze(ByVal dwFreeze As Integer) As Integer  
Implements UnsafeNativeMethods.IViewObject.Unfreeze

[JScript] function UnsafeNativeMethods.IViewObject.Unfreeze(dwFreeze : int) :  
int;

**kkkkkkkkkkkkkkkkkk)UnsafeNativeMethods.IViewObject2.Draw**

[C#] void UnsafeNativeMethods.IViewObject2.Draw(int dwDrawAspect, int  
lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE ptd, IntPtr  
hdcTargetDev, IntPtr hdcDraw, NativeMethods.COMRECT lprcBounds,  
NativeMethods.COMRECT lprcWBounds, IntPtr pfnContinue, int dwContinue);

```

1 [C++] void UnsafeNativeMethods::IViewObject2::Draw(int dwDrawAspect, int
2 lindex, IntPtr pvAspect, NativeMethods.tagDVTARGETDEVICE* ptd, IntPtr
3 hdcTargetDev, IntPtr hdcDraw, NativeMethods.COMRECT* lprcBounds,
4 NativeMethods.COMRECT* lprcWBounds, IntPtr pfnContinue, int dwContinue);
5 [VB] Sub IViewObject2.Draw(ByVal dwDrawAspect As Integer, ByVal lindex
6 As Integer, ByVal pvAspect As IntPtr, ByVal ptd As
7 NativeMethods.tagDVTARGETDEVICE, ByVal hdcTargetDev As IntPtr, ByVal
8 hdcDraw As IntPtr, ByVal lprcBounds As NativeMethods.COMRECT, ByVal
9 lprcWBounds As NativeMethods.COMRECT, ByVal pfnContinue As IntPtr,
10 ByVal dwContinue As Integer) Implements
11 UnsafeNativeMethods.IViewObject2.Draw
12 [JScript] function UnsafeNativeMethods.IViewObject2.Draw(dwDrawAspect :
13 int, lindex : int, pvAspect : IntPtr, ptd : NativeMethods.tagDVTARGETDEVICE,
14 hdcTargetDev : IntPtr, hdcDraw : IntPtr, lprcBounds :
15 NativeMethods.COMRECT, lprcWBounds : NativeMethods.COMRECT,
16 pfnContinue : IntPtr, dwContinue : int);
17 !!!!!!!!!!!!!!UnsafeNativeMethods.IViewObject2 Freeze
18
19 [C#] int UnsafeNativeMethods.IViewObject2.Freeze(int dwDrawAspect, int
20 lindex, IntPtr pvAspect, IntPtr pdwFreeze);
21 [C++] int UnsafeNativeMethods::IViewObject2::Freeze(int dwDrawAspect, int
22 lindex, IntPtr pvAspect, IntPtr pdwFreeze);
23 [VB] Function IViewObject2.Freeze(ByVal dwDrawAspect As Integer, ByVal
24 lindex As Integer, ByVal pvAspect As IntPtr, ByVal pdwFreeze As IntPtr) As
25 Integer Implements UnsafeNativeMethods.IViewObject2.Freeze

```



1 ppColorSet As NativeMethods.tagLOGPALETTE) As Integer Implements  
2 UnsafeNativeMethods.IViewObject2.GetColorSet

3 [JScript] function

4 UnsafeNativeMethods.IViewObject2.GetColorSet(dwDrawAspect : int, lindex :  
5 int, pvAspect : IntPtr, ptd : NativeMethods.tagDVTARGETDEVICE,  
6 hicTargetDev : IntPtr, ppColorSet : NativeMethods.tagLOGPALETTE) : int;

7 *oooooooooooooooooooo)UnsafeNativeMethods.IViewObject2.GetExtent*

8  
9 [C#] void UnsafeNativeMethods.IViewObject2.GetExtent(int dwDrawAspect, int  
10 lindex, NativeMethods.tagDVTARGETDEVICE ptd, NativeMethods.tagSIZEL  
11 lpsize);

12 [C++] void UnsafeNativeMethods::IViewObject2::GetExtent(int dwDrawAspect,  
13 int lindex, NativeMethods.tagDVTARGETDEVICE\* ptd,  
14 NativeMethods.tagSIZEL\* lpsize);

15 [VB] Sub IViewObject2.GetExtent(ByVal dwDrawAspect As Integer, ByVal  
16 lindex As Integer, ByVal ptd As NativeMethods.tagDVTARGETDEVICE, ByVal  
17 lpsize As NativeMethods.tagSIZEL) Implements

18 UnsafeNativeMethods.IViewObject2.GetExtent

19 [JScript] function UnsafeNativeMethods.IViewObject2.GetExtent(dwDrawAspect  
20 : int, lindex : int, ptd : NativeMethods.tagDVTARGETDEVICE, lpsize :  
21 NativeMethods.tagSIZEL);

22 *pppppppppppppppppppp)UnsafeNativeMethods.IViewObject2.SetAdvise*

23  
24 [C#] void UnsafeNativeMethods.IViewObject2.SetAdvise(int aspects, int advf,  
25

UnsafeNativeMethods.IAdviseSink pAdvSink);

[C++] void UnsafeNativeMethods::IViewObject2::SetAdvise(int aspects, int advf,

UnsafeNativeMethods.IAdviseSink\* pAdvSink);

[VB] Sub IViewObject2.SetAdvise(ByVal aspects As Integer, ByVal advf As

Integer, ByVal pAdvSink As UnsafeNativeMethods.IAdviseSink) Implements

UnsafeNativeMethods.IViewObject2.SetAdvise

[JScript] function UnsafeNativeMethods.IViewObject2.SetAdvise(aspects : int,

advf : int, pAdvSink : UnsafeNativeMethods.IAdviseSink);

*qqqqqqqqqqqqqqqqqq)UnsafeNativeMethods.IViewObject2.Unfreeze*

[C#] int UnsafeNativeMethods.IViewObject2.Unfreeze(int dwFreeze);

[C++] int UnsafeNativeMethods::IViewObject2::Unfreeze(int dwFreeze);

[VB] Function IViewObject2.Unfreeze(ByVal dwFreeze As Integer) As Integer

Implements UnsafeNativeMethods.IViewObject2.Unfreeze

[JScript] function UnsafeNativeMethods.IViewObject2.Unfreeze(dwFreeze : int) :

int;

*rrrrrrrrrrrrrrrrrrrr)Update*

[C#] public void Update();

[C++] public: void Update();

[VB] Public Sub Update()

[JScript] public function Update();

#### *Description*

Causes the control to redraw the invalidated regions with its client area.

## ssssssssssssssssssss)UpdateBounds

[C#]                   protected                   void                   UpdateBounds();  
[C++]                   protected:                   void                   UpdateBounds();  
[VB]                   Protected                   Sub                   UpdateBounds()  
[JScript] protected function UpdateBounds(); Updates the bounds of the control.

### Description

Updates the bounds of the control with the current size and location.

If the new **System.Windows.Forms.Control.Size** of the control is different from the previous **System.Drawing.Size**, the **System.Windows.Forms.Control.SizeChanged** event is raised. Likewise, if the **System.Windows.Forms.Control.Location** of the control changes, the **System.Windows.Forms.Control.LocationChanged** event is raised.

## tttttttttttttttttttt)UpdateBounds

[C#]   protected   void   UpdateBounds(int x, int y, int width, int height);  
[C++]   protected:   void   UpdateBounds(int x, int y, int width, int height);  
[VB]   Protected   Sub   UpdateBounds(ByVal x As Integer, ByVal y As Integer,  
ByVal   width   As   Integer,   ByVal   height   As   Integer)  
[JScript] protected function UpdateBounds(x : int, y : int, width : int, height : int);

### Description

Updates the bounds of the control with the specified size and location.

If the new **System.Windows.Forms.Control.Size** of the control is different from the previous **System.Drawing.Size**, the **System.Windows.Forms.Control.SizeChanged** event is raised. Likewise, if the **System.Windows.Forms.Control.Location** of the control changes, the **System.Windows.Forms.Control.LocationChanged** event is raised. The

**System.Drawing.Point.X** coordinate of the control. The  
**System.Drawing.Point.Y** coordinate of the control. The  
**System.Drawing.Size.Width** of the control. The  
**System.Drawing.Size.Height** of the control.

[illegible]

```
[C#] protected void UpdateBounds(int x, int y, int width, int height, int
clientWidth, int clientHeight);
```

```
[C++] protected: void UpdateBounds(int x, int y, int width, int height, int
clientWidth, int clientHeight);
```

```
[VB] Protected Sub UpdateBounds(ByVal x As Integer, ByVal y As Integer,
ByVal width As Integer, ByVal height As Integer, ByVal clientWidth As Integer,
ByVal clientHeight As Integer)
```

```
[JScript] protected function UpdateBounds(x : int, y : int, width : int, height : int,
clientWidth      :      int,      clientHeight      :      int);
```

*Description*

Updates the bounds of the control with the specified size, location, and client size.

If the new **System.Windows.Forms.Control.Size** of the control is different from the previous **System.Drawing.Size**, the **System.Windows.Forms.Control.SizeChanged** event is raised. Likewise, if the **System.Windows.Forms.Control.Location** of the control changes, the **System.Windows.Forms.Control.LocationChanged** event is raised. The **System.Drawing.Point.X** coordinate of the control. The **System.Drawing.Point.Y** coordinate of the control. The **System.Drawing.Size.Width** of the control. The **System.Drawing.Size.Height** of the control. The client **System.Drawing.Size.Width** of the control. The client **System.Drawing.Size.Height** of the control.



## UpdateStyles

|           |            |          |                 |
|-----------|------------|----------|-----------------|
| [C#]      | protected  | void     | UpdateStyles(); |
| [C++]     | protected: | void     | UpdateStyles(); |
| [VB]      | Protected  | Sub      | UpdateStyles()  |
| [JScript] | protected  | function | UpdateStyles(); |

### Description

Forces the assigned styles to be reapplied to the control.

This method calls the **System.Windows.Forms.Control.CreateParams** method to get the styles to apply. The styles assigned to the **System.Windows.Forms.CreateParams.Style** and **System.Windows.Forms.CreateParams.ExStyle** properties of the **System.Windows.Forms.CreateParams** object assigned to the control's **System.Windows.Forms.Control.CreateParams** property are reapplied. The control is repainted to reflect the style changes if necessary.

## UpdateZOrder

|           |            |          |                 |
|-----------|------------|----------|-----------------|
| [C#]      | protected  | void     | UpdateZOrder(); |
| [C++]     | protected: | void     | UpdateZOrder(); |
| [VB]      | Protected  | Sub      | UpdateZOrder()  |
| [JScript] | protected  | function | UpdateZOrder(); |

### Description

Updates control in its parent's z-order.

The **System.Windows.Forms.Control.UpdateZOrder** method updates the position of the control in its parent control's z-order. For example, if this control is a newly created control that was added to a **System.Windows.Forms.Control.ControlCollection**, the z-order is updated with the new control added to the bottom.

## xxxxxxxxxxxxxxxxxxxx)WndProc

[C#]      protected      virtual      void      WndProc(ref      Message      m);  
[C++]      protected:      virtual      void      WndProc(Message\*      m);  
[VB]      Overridable      Protected      Sub      WndProc(ByRef      m      As      Message)  
[JScript]      protected      function      WndProc(m      :      Message);

### Description

Processes Windows messages.

All messages are sent to the

**System.Windows.Forms.Control.WndProc(System.Windows.Forms.Message@)** method after getting filtered through the **System.Windows.Forms.Control.PreProcessMessage(System.Windows.Forms.Message@)** method. The Windows **System.Windows.Forms.Message** to process.

Control.ControlAccessibleObject class (System.Windows.Forms)

### a)      *WndProc*

### Description

Provides information about a control so it can be inspected by an accessibility application.

Windows Forms has accessibility support built in, and provides information about your application which allows it to work with accessibility client applications.

Examples of accessibility client applications are: screen enlarger and reviewer utilities, voice input utilities, on-screen keyboards, alternative input devices, and keyboard enhancement utilities. There are instances when you will want to provide additional information to accessibility client applications. There are two ways of providing this additional information. To provide limited accessibility information for existing controls, set the

**System.Windows.Forms.Control.AccessibleName** ,  
**System.Windows.Forms.Control.AccessibleDescription** ,

**System.Windows.Forms.Control.AccessibleDefaultActionDescription** , and **System.Windows.Forms.Control.AccessibleRole** property values of the control, which will be reported to accessibility client applications. Alternatively, if you require more accessibility information to be included with your control, you can write your own class deriving from the **System.Windows.Forms.AccessibleObject** or **System.Windows.Forms.Control.ControlAccessibleObject** classes. For example, if you are writing your own control that is not derived from the common controls or you require such operations as hit testing within your control, you should create a **System.Windows.Forms.Control.ControlAccessibleObject** for your control. You can create a **System.Windows.Forms.Control.ControlAccessibleObject** for your control by calling the **System.Windows.Forms.Control.CreateAccessibilityInstance** method.

*b) Control.ControlAccessibleObject*

*Example Syntax:*

*c) WndProc*

```
[C#]    public    Control.ControlAccessibleObject(Control    ownerControl);
[C++]    public:    ControlAccessibleObject(Control*    ownerControl);
[VB]    Public    Sub    New(ByVal    ownerControl    As    Control)
[JScript]    public    function    Control.ControlAccessibleObject(ownerControl    :
Control);
```

### *Description*

Initializes a new instance of the **System.Windows.Forms.Control.ControlAccessibleObject** class. The **System.Windows.Forms.Control** that owns the **System.Windows.Forms.Control.ControlAccessibleObject**.

- d) *Bounds*
- e) *DefaultAction*
- f) *WndProc*

#### *Description*

Gets the default action of the  
**System.Windows.Forms.Control.ControlAccessibleObject** .

- g) *Description*
- h) *WndProc*

```
[C#]      public      override      string      Description      {get;}
[C++]     public:      __property      virtual      String*      get_Description();
[VB]      Overrides      Public      ReadOnly      Property      Description      As      String
[JScript]      public      function      get      Description()      :      String;
```

#### *Description*

Gets the description of the  
**System.Windows.Forms.Control.ControlAccessibleObject** .

- i) *Handle*
- j) *WndProc*

```
[C#]      public      IntPtr      Handle      {get;      set;}
[C++]     public:      __property      IntPtr      get_Handle();public:      __property      void
set_Handle(IntPtr);
```

[VB]            Public            Property            Handle            As            IntPtr

[JScript] public function get Handle() : IntPtr; public function set Handle(IntPtr);

#### *Description*

Gets or sets the handle of the

**System.Windows.Forms.Control.ControlAccessibleObject** .

The value of the

**System.Windows.Forms.Control.ControlAccessibleObject.Handle** property for the **System.Windows.Forms.Control.ControlAccessibleObject** is equal to the **System.Windows.Forms.Control.Handle** property of the **System.Windows.Forms.Control** it is associated with.

*k)      Help*

*l)      WndProc*

[C#]            public            override            string            Help            {get;}

[C++]           public:           \_\_property           virtual           String\*           get\_Help();

[VB]   Overrides   Public   ReadOnly   Property   Help   As   String

[JScript]       public           function           get           Help()           :           String;

#### *Description*

Gets the description of what the object does or how the object is used.

*m)      KeyboardShortcut*

*n)      WndProc*

[C#]            public            override            string            KeyboardShortcut            {get;}

[C++]           public:           \_\_property           virtual           String\*           get\_KeyboardShortcut();

[VB]   Overrides   Public   ReadOnly   Property   KeyboardShortcut   As   String

1 [JScript] public function get KeyboardShortcut() : String;

3 *Description*

4 Gets the object shortcut key or access key for an accessible object.

5 All objects that have a shortcut key or access key should support this property.

6 o) *Name*

7 p) *WndProc*

9 [C#] public override string Name {get; set;}

10 [C++] public: \_\_property virtual String\* get\_Name();public: \_\_property virtual

11 void set\_Name(String\*);

12 [VB] Overrides Public Property Name As String

13 [JScript] public function get Name() : String;public function set Name(String);

15 *Description*

16 Gets or sets the accessible object name.

17 q) *Owner*

18 r) *WndProc*

20 [C#] public Control Owner {get;}

21 [C++] public: \_\_property Control\* get\_Owner();

22 [VB] Public ReadOnly Property Owner As Control

23 [JScript] public function get Owner() : Control;

*Description*

Gets the owner of the  
**System.Windows.Forms.Control.ControlAccessibleObject** .

- s) Parent*
- t) Role*
- u) WndProc*

*Description*

Gets the role of this accessible object.

- v) State*
- w) Value*
- x) GetHelpTopic*

[C#] public override int GetHelpTopic(out string fileName);

[C++] public: int GetHelpTopic(String\*\* fileName);

[VB] Overrides Public Function GetHelpTopic(ByRef fileName As String) As Integer

[JScript] public override function GetHelpTopic(fileName : String) : int;

*Description*

Gets an identifier for a Help topic and the path to the Help file associated with this accessible object.

*Return Value:* An identifier for a Help topic, or -1 if there is no Help topic. On return, the *fileName* parameter will contain the path to the Help file associated with this accessible object, or **null** if there is no **IAccessible** interface specified.

Pass the identifier to the WinHelp file specified by the *fileName* parameter to identify the desired Help topic. When this method returns, contains a string that represents the path to the Help file associated with this accessible object. This parameter is passed uninitialized.

y) *NotifyClients*

[C#] public void NotifyClients(AccessibleEvents accEvent);

[C++] public: void NotifyClients(AccessibleEvents accEvent);

[VB] Public Sub NotifyClients(ByVal accEvent As AccessibleEvents)

[JScript] public function NotifyClients(accEvent : AccessibleEvents); Notifies accessibility client applications of **System.Windows.Forms.AccessibleEvents** .

*Description*

Notifies accessibility client applications of the specified **System.Windows.Forms.AccessibleEvents** .

You must call the

**System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method for each **System.Windows.Forms.AccessibleEvents** object the accessibility client applications are to be notified of. The

**System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method is typically called when a property is set or from within an event handler. For example, you might call the

**System.Windows.Forms.Control.ControlAccessibleObject.NotifyEvents** method and pass in **System.Windows.Forms.AccessibleEvents.Hide** from within the event handler for the

**System.Windows.Forms.Control.VisibleChanged** event. The **System.Windows.Forms.AccessibleEvents** object to notify the accessibility client applications of.

z) *NotifyClients*

[C#] public void NotifyClients(AccessibleEvents accEvent, int childID);



```

1  [C++] public: void NotifyClients(AccessibleEvents accEvent, int childID);
2  [VB] Public Sub NotifyClients(ByVal accEvent As AccessibleEvents, ByVal
3  childID                                     As                                     Integer)
4  [JScript] public function NotifyClients(accEvent : AccessibleEvents, childID :
5  int);

```

### Description

Notifies the accessibility client applications of the specified **System.Windows.Forms.AccessibleEvents** for the specified child control.

You must call the

**System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method for each **System.Windows.Forms.AccessibleEvents** object the accessibility client applications are to be notified of. The **System.Windows.Forms.Control.ControlAccessibleObject.NotifyClients(System.Windows.Forms.AccessibleEvents)** method is typically called when a property is set or from within an event handler. For example, you might call the **System.Windows.Forms.Control.ControlAccessibleObject.NotifyEvents** method and pass in **System.Windows.Forms.AccessibleEvents.Hide** from within the event handler for the **System.Windows.Forms.Control.VisibleChanged** event. The **System.Windows.Forms.AccessibleEvents** object to notify the accessibility client applications of. The child **System.Windows.Forms.Control** to notify of the accessible event.

#### aa) ToString

```

21  [C#]          public          override          string          ToString();
22  [C++]          public:          String*          ToString();
23  [VB]  Overrides  Public  Function  ToString()  As  String
24  [JScript]  public  override  function  ToString()  :  String;

```

*Description*

ControlBindingsCollection class (System.Windows.Forms)

*a) UseStdAccessibleObjects*

*Description*

Represents the collection of data bindings for a control.

Simple data binding is accomplished by adding **System.Windows.Forms.Binding** objects to a **System.Windows.Forms.ControlBindingsCollection**. Any object that inherits from the **System.Windows.Forms.Control** class can access the **System.Windows.Forms.ControlBindingsCollection** through the **System.Windows.Forms.Control.DataBindings** property. For a list of Windows controls that support data binding, see the **System.Windows.Forms.Binding** class.

*b) Control*

*c) UseStdAccessibleObjects*

|           |         |            |          |                      |
|-----------|---------|------------|----------|----------------------|
| [C#]      | public  | Control    | Control  | {get;}               |
| [C++]     | public: | __property | Control* | get_Control();       |
| [VB]      | Public  | ReadOnly   | Property | Control As Control   |
| [JScript] | public  | function   | get      | Control() : Control; |

*Description*

Gets the control that the collection belongs to.

- d) *Count*
- e) *IsReadOnly*
- f) *IsSynchronized*
- g) *Item*
- h) *UseStdAccessibleObjects*

## **System.Windows.Forms.Binding**

### *Description*

Gets the **System.Windows.Forms.Binding** specified by the control's property name.

```
private void PrintValue() { ControlBindingsCollection myControlBindings;
myControlBindings = textBox1.DataBindings; // Get the Binding for the Text
property. Binding myBinding = myControlBindings["Text"]; // Assuming the data
source is a DataTable. DataRowView drv; drv =
(DataRowView)myBinding.BindingManagerBase.Current; // Assuming a column
named "custName" exists, print the value. Console.WriteLine(drv["custName"]);
} The following example uses the
System.Windows.Forms.ControlBindingsCollection.Item(System.String
) property to return the System.Windows.Forms.Binding for a
System.Windows.Forms.TextBox control's
System.Windows.Forms.Control.Text property. The name of the property
on the data-bound control.
```

- i) *Item*
- j) *List*
- k) *SyncRoot*

## **System.Windows.Forms.Binding**

### *Description*

Adds the specified **System.Windows.Forms.Binding** to the collection.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs when the change is complete. The **System.Windows.Forms.Binding** to add.

*l) Add*

[C#] public void Add(Binding binding);

[C++] public: void Add(Binding\* binding);

[VB] Public Sub Add(ByVal binding As Binding)

[JScript] public function Add(binding : Binding); Adds a

**System.Windows.Forms.Binding** to the collection.

*Description*

Adds the specified **System.Windows.Forms.Binding** to the collection.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs when the change is complete. The **System.Windows.Forms.Binding** to add.

*m) Add*

[C#] public Binding Add(string propertyName, object dataSource, string dataMember);

[C++] public: Binding\* Add(String\* propertyName, Object\* dataSource, String\* dataMember);

[VB] Public Function Add(ByVal propertyName As String, ByVal dataSource As

Object, ByVal dataMember As String) As Binding

[JScript] public function Add(propertyName : String, dataSource : Object,

dataMember : String) : Binding;

## Description

Creates a **System.Windows.Forms.Binding** using the specified control property name, data source, and data member, and adds it to the collection.

*Return Value:* The newly-created **System.Windows.Forms.Binding**.

Adding a **System.Windows.Forms.Binding** causes the **System.Windows.Forms.BindingsCollection.CollectionChanged** event to occur. The name of the control property to bind. An **System.Object** that represents the data source. The property or list to bind to.

### n) AddCore

[C#]      protected      override      void      AddCore(Binding      dataBinding);

[C++]      protected:      void      AddCore(Binding\*      dataBinding);

[VB]      Overrides      Protected      Sub      AddCore(ByVal      dataBinding      As      Binding)

[JScript]      protected      override      function      AddCore(dataBinding      :      Binding);

## Description

Adds a binding to the collection. The **System.Windows.Forms.Binding** to add.

## Description

Clears the collection of any bindings.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs when the collection is cleared.

### o) Clear

[C#]                      public                      void                      Clear();

[C++]                      public:                      void                      Clear();

[VB]                      Public                      Sub                      Clear()

[JScript] public function Clear(); Clears the collection of any bindings.

### *Description*

Clears the collection of any bindings.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs when the collection is cleared.

### *p) ClearCore*

|           |            |           |          |              |
|-----------|------------|-----------|----------|--------------|
| [C#]      | protected  | override  | void     | ClearCore(); |
| [C++]     | protected: |           | void     | ClearCore(); |
| [VB]      | Overrides  | Protected | Sub      | ClearCore()  |
| [JScript] | protected  | override  | function | ClearCore(); |

### *Description*

## **System.Windows.Forms.Binding**

### *Description*

Deletes the specified **System.Windows.Forms.Binding** from the collection.  
The **System.Windows.Forms.Binding** to remove.

### *q) Remove*

|                                     |                 |                            |                       |                     |
|-------------------------------------|-----------------|----------------------------|-----------------------|---------------------|
| [C#]                                | public          | void                       | Remove(Binding        | binding);           |
| [C++]                               | public:         | void                       | Remove(Binding*       | binding);           |
| [VB]                                | Public          | Sub                        | Remove(ByVal          | binding As Binding) |
| [JScript]                           | public function | Remove(binding : Binding); | Deletes the specified |                     |
| <b>System.Windows.Forms.Binding</b> |                 | from                       | the                   | collection.         |

*Description*

Deletes the specified **System.Windows.Forms.Binding** from the collection.  
The **System.Windows.Forms.Binding** to remove.

*Description*

Deletes the **System.Windows.Forms.Binding** at the specified index.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs if the removal succeeds. The zero-based index of the item to remove.

*r) RemoveAt*

|           |         |          |                |                   |
|-----------|---------|----------|----------------|-------------------|
| [C#]      | public  | void     | RemoveAt(int   | index);           |
| [C++]     | public: | void     | RemoveAt(int   | index);           |
| [VB]      | Public  | Sub      | RemoveAt(ByVal | index As Integer) |
| [JScript] | public  | function | RemoveAt(index | : int);           |

*Description*

Deletes the **System.Windows.Forms.Binding** at the specified index.

The **System.Windows.Forms.BindingsCollection.CollectionChanged** event occurs if the removal succeeds. The zero-based index of the item to remove.

*s) RemoveCore*

|           |            |           |                     |                        |                         |
|-----------|------------|-----------|---------------------|------------------------|-------------------------|
| [C#]      | protected  | override  | void                | RemoveCore(Binding     | dataBinding);           |
| [C++]     | protected: | void      | RemoveCore(Binding* | dataBinding);          |                         |
| [VB]      | Overrides  | Protected | Sub                 | RemoveCore(ByVal       | dataBinding As Binding) |
| [JScript] | protected  | override  | function            | RemoveCore(dataBinding | : Binding);             |

*Description*

TabControl.ControlCollection class (System.Windows.Forms)

*a) ToString*

*Description*

Contains a collection of **System.Windows.Forms.Control** objects.

*b) TabControl.ControlCollection*

*Example Syntax:*

*c) ToString*

[C#]        public        TabControl.ControlCollection(TabControl        owner);

[C++]        public:        ControlCollection(TabControl\*        owner);

[VB]        Public        Sub        New(ByVal        owner        As        TabControl)

[JScript]   public   function   TabControl.ControlCollection(owner : TabControl);

*Description*

Initializes a new instance of the **System.Windows.Forms.TabControl.ControlCollection** class. The **System.Windows.Forms.TabControl** that this collection belongs to.



d) *Count*

e) *IsReadOnly*

f) *Item*

g) *Add*

[C#] public override void Add(Control value);

[C++] public: void Add(Control\* value);

[VB] Overrides Public Sub Add(ByVal value As Control)

[JScript] public override function Add(value : Control);

#### *Description*

Adds a **System.Windows.Forms.Control** to the collection.

You cannot add a **System.Windows.Forms.TabPage** to a **System.Windows.Forms.TabPage**. The **System.Windows.Forms.Control** to add.

h) *Remove*

[C#] public override void Remove(Control value);

[C++] public: void Remove(Control\* value);

[VB] Overrides Public Sub Remove(ByVal value As Control)

[JScript] public override function Remove(value : Control);

#### *Description*

Removes a **System.Windows.Forms.Control** from the collection. The **System.Windows.Forms.Control** to remove.

Control.ControlCollection class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a collection of **System.Windows.Forms.Control** objects

The

**System.Windows.Forms.Control.ControlCollection.Add(System.Windows.Forms.Control)** ,

**System.Windows.Forms.Control.ControlCollection.Remove(System.Windows.Forms.Control)** , and

**System.Windows.Forms.Control.ControlCollection.RemoveAt(System.Int32)** methods enable you to add and remove individual controls from the collection. You can also use the

**System.Windows.Forms.Control.ControlCollection.AddRange(System.Windows.Forms.Control[])** or

**System.Windows.Forms.Control.ControlCollection.Clear** methods to add or remove all the controls from the collection.

b) *Control.ControlCollection*

*Example Syntax:*

c) *ToString*

[C#]            public            Control.ControlCollection(Control            owner);

[C++]            public:            ControlCollection(Control\*            owner);

[VB]            Public            Sub            New(ByVal            owner            As            Control)

[JScript]            public            function            Control.ControlCollection(owner            :            Control);

*Description*

Initializes a new instance of the

**System.Windows.Forms.Control.ControlCollection** class. A

**System.Windows.Forms.Control** representing the control that owns the control collection.

d) *Count*

e) *ToString*

[C#]                    public                    int                    Count                    {get;}

[C++]                    public:                    \_\_property                    int                    get\_Count();

[VB]                    Public                    ReadOnly                    Property                    Count                    As                    Integer

[JScript]                    public                    function                    get                    Count()                    :                    int;

#### *Description*

Gets the total number of **System.Windows.Forms.Control** objects in the collection.

The **System.Windows.Forms.Control.ControlCollection.Count** property holds the number of **System.Windows.Forms.Control** objects assigned to the collection. You can use the

**System.Windows.Forms.Control.ControlCollection.Count** property value as the upper bounds of a loop to iterate through a collection.

f) *IsReadOnly*

g) *ToString*

[C#]                    public                    bool                    IsReadOnly                    {get;}

[C++]                    public:                    \_\_property                    bool                    get\_IsReadOnly();

[VB]                    Public                    ReadOnly                    Property                    IsReadOnly                    As                    Boolean

[JScript]                    public                    function                    get                    IsReadOnly()                    :                    Boolean;

#### *Description*

Gets a value indicating whether the control collection is read-only.

h) *Item*

i) *ToString*

```
[C#]      public      virtual      Control      this[int      index]      {get;}
[C++]      public:      __property      virtual      Control*      get_Item(int      index);
[VB]      Overridable      Public      Default      ReadOnly      Property      Item(ByVal      index      As
Integer)
As
Control
[JScript]      returnValue      =      ControlCollectionObject.Item(index);
```

*Description*

Indicates the **System.Windows.Forms.Control** at the specified indexed location in the collection.

To retrieve a **System.Windows.Forms.Control** from the **System.Windows.Forms.Control.ControlCollection**, reference the collection object with a specific index value. The index value of the **System.Windows.Forms.Control.ControlCollection** is a zero-based index. The index of the control to retrieve from the control collection.

j) *Add*

```
[C#]      public      virtual      void      Add(Control      value);
[C++]      public:      virtual      void      Add(Control*      value);
[VB]      Overridable      Public      Sub      Add(ByVal      value      As      Control)
[JScript]      public      function      Add(value      :      Control);
```

*Description*

Adds the specified control to the control collection.

The  
**System.Windows.Forms.Control.ControlCollection.Add(System.Window**

**s.Forms.Control**) method allows you to add **System.Windows.Forms.Control** objects to the end of the control collection. The **System.Windows.Forms.Control** to add to the control collection.

*k) AddRange*

```
[C#]      public      virtual      void      AddRange(Control[]      controls);
[C++]     public:     virtual      void      AddRange(Control*      controls[]);
[VB]      Overridable Public Sub      AddRange(ByVal controls() As Control)
[JScript]      public      function      AddRange(controls      :      Control[]);
```

*Description*

Adds an array of control objects to the collection.

The **System.Windows.Forms.Control** objects contained in the *controls* array are appended to the end of the collection. An array of **System.Windows.Forms.Control** objects to add to the collection.

*l) Clear*

```
[C#]      public      virtual      void      Clear();
[C++]     public:     virtual      void      Clear();
[VB]      Overridable      Public      Sub      Clear()
[JScript]      public      function      Clear();
```

*Description*

Removes all controls from the collection.

You can use the **System.Windows.Forms.Control.ControlCollection.Clear** method to remove the entire collection of controls from a parent control.

### *m) Contains*

```
[C#]      public      bool      Contains(Control      control);
[C++]      public:      bool      Contains(Control*      control);
[VB] Public Function Contains(ByVal control As Control) As Boolean
[JScript] public function Contains(control : Control) : Boolean;
```

#### *Description*

Determines whether the specified control is a member of the collection.

*Return Value:* **true** if the **System.Windows.Forms.Control** is a member of the collection; otherwise, **false** .

This method enables you to determine whether a **System.Windows.Forms.Control** is a member of the collection before attempting to perform operations on the **System.Windows.Forms.Control** . You can use this method to confirm that a **System.Windows.Forms.Control** has been added to or is still a member of the collection. The **System.Windows.Forms.Control** to locate in the collection.

### *n) CopyTo*

```
[C#]      public      void      CopyTo(Array      dest,      int      index);
[C++]      public:      __sealed      void      CopyTo(Array*      dest,      int      index);
[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)
[JScript] public function CopyTo(dest : Array, index : int);
```

#### *Description*

Copies the entire collection into an existing array at a specified location within the array.

All the controls in the collection are copied into the array starting at the specified indexed location, overwriting any existing data within the range of objects copied into the array. The destination array. The index in the destination array at which storing begins.

*o) Equals*

[C#] public override bool Equals(object other);

[C++] public: bool Equals(Object\* other);

[VB] Overrides Public Function Equals(ByVal other As Object) As Boolean

[JScript] public override function Equals(other : Object) : Boolean;

*Description*

*p) GetChildIndex*

[C#] public int GetChildIndex(Control child);

[C++] public: int GetChildIndex(Control\* child);

[VB] Public Function GetChildIndex(ByVal child As Control) As Integer

[JScript] public function GetChildIndex(child : Control) : int; Retrieves the index of a control within the control collection.

*Description*

Retrieves the index of the specified child control within the control collection.

*Return Value:* A zero-based index value that represents the location of the specified child control within the control collection.

The control with an index value of zero is at the top of the z-order, and higher numbers are closer to the bottom. The **System.Windows.Forms.Control** to search for in the control collection.

q) *GetChildIndex*

```
[C#] public int GetChildIndex(Control child, bool throwException);  
[C++] public: int GetChildIndex(Control* child, bool throwException);  
[VB] Public Function GetChildIndex(ByVal child As Control, ByVal  
throwException As Boolean) As Integer  
[JScript] public function GetChildIndex(child : Control, throwException :  
Boolean) : int;
```

*Description*

Retrieves the index of the specified child control within the control collection, and optionally raises an exception if the specified control is not within the control collection.

*Return Value:* A zero-based index value that represents the location of the specified child control within the control collection.

The control with an index value of zero is at the top of the z-order, and higher numbers are closer to the bottom. The **System.Windows.Forms.Control** to search for in the control collection. **true** to throw an exception if the **System.Windows.Forms.Control** specified in the *child* parameter is not a control in the **System.Windows.Forms.Control.ControlCollection** ; otherwise, **false** .

r) *GetEnumerator*

```
[C#] public IEnumerator GetEnumerator();  
[C++] public: __sealed IEnumerator* GetEnumerator();  
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator  
[JScript] public function GetEnumerator() : IEnumerator;
```

*Description*



Returns an enumerator that can be used to iterate through the control collection.  
*Return Value:* An **System.Collections.IEnumerator** object that represents the control collection.

s) *GetHashCode*

|           |           |          |          |                          |
|-----------|-----------|----------|----------|--------------------------|
| [C#]      | public    | override | int      | GetHashCode();           |
| [C++]     | public:   |          | int      | GetHashCode();           |
| [VB]      | Overrides | Public   | Function | GetHashCode() As Integer |
| [JScript] | public    | override | function | GetHashCode() : int;     |

*Description*

t) *IndexOf*

|           |         |          |                                   |            |
|-----------|---------|----------|-----------------------------------|------------|
| [C#]      | public  | int      | IndexOf(Control                   | control);  |
| [C++]     | public: | int      | IndexOf(Control*                  | control);  |
| [VB]      | Public  | Function | IndexOf(ByVal control As Control) | As Integer |
| [JScript] | public  | function | IndexOf(control : Control)        | : int;     |

*Description*

Retrieves the index of the specified control in the control collection.  
*Return Value:* A zero-based index value that represents the position of the specified **System.Windows.Forms.Control** in the **System.Windows.Forms.Control.ControlCollection** .

If the control is not found in the collection, the **System.Windows.Forms.Control.ControlCollection.IndexOf(System.Windows.Forms.Control)** method return value is -1. The **System.Windows.Forms.Control** to locate in the collection.

u) *Remove*

```
[C#]      public      virtual      void      Remove(Control      value);
[C++]     public:     virtual      void      Remove(Control*      value);
[VB]      Overridable Public Sub Remove(ByVal value As Control)
[JScript] public      function      Remove(value      :      Control);
```

*Description*

Removes the specified control from the control collection.

When a **System.Windows.Forms.Control** is removed from the control collection, all subsequent controls are moved up one position in the collection. The **System.Windows.Forms.Control** to remove from the **System.Windows.Forms.Control.ControlCollection**.

v) *RemoveAt*

```
[C#]      public      void      RemoveAt(int      index);
[C++]     public:     __sealed      void      RemoveAt(int      index);
[VB]      NotOverridable Public Sub RemoveAt(ByVal index As Integer)
[JScript] public      function      RemoveAt(index      :      int);
```

*Description*

Removes a control from the control collection at the specified indexed location.

When a **System.Windows.Forms.Control** is removed from the control collection, all subsequent controls are moved up one position in the collection. The index value of the **System.Windows.Forms.Control** to remove.

w) *SetChildIndex*

```
[C#]    public    void    SetChildIndex(Control    child,    int    newIndex);  
[C++]    public:    void    SetChildIndex(Control*    child,    int    newIndex);  
[VB]    Public Sub SetChildIndex(ByVal child As Control, ByVal newIndex As  
Integer)  
[JScript]    public    function    SetChildIndex(child : Control, newIndex : int);
```

*Description*

Sets the index of the specified child control in the collection to the specified index value.

The control with an index value of zero is at the top of the z-order, and higher numbers are closer to the bottom. The child **System.Windows.Forms.Control** to search for. The new index value of the control.

x) *IList.Add*

```
[C#]                int                IList.Add(object                control);  
[C++]                int                IList::Add(Object*                control);  
[VB]    Function Add(ByVal control As Object) As Integer Implements IList.Add  
[JScript]    function IList.Add(control : Object) : int;
```

y) *IList.Contains*

```
[C#]                bool                IList.Contains(object                control);  
[C++]                bool                IList::Contains(Object*                control);  
[VB]    Function Contains(ByVal control As Object) As Boolean Implements
```

1 IList.Contains

2 [JScript] function IList.Contains(control : Object) : Boolean;

3       **z) IList.IndexOf**

4  
5 [C#]               int               IList.IndexOf(object               control);

6 [C++]               int               IList::IndexOf(Object\*               control);

7 [VB] Function IndexOf(ByVal control As Object) As Integer Implements  
8 IList.IndexOf

9 [JScript] function IList.IndexOf(control : Object) : int;

10       **aa) IList.Insert**

11  
12 [C#]               void               IList.Insert(int               index,               object               value);

13 [C++]               void               IList::Insert(int               index,               Object\*               value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements  
15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17       **bb) IList.Remove**

18  
19 [C#]               void               IList.Remove(object               control);

20 [C++]               void               IList::Remove(Object\*               control);

21 [VB] Sub Remove(ByVal control As Object) Implements IList.Remove

22 [JScript] function IList.Remove(control : Object);

**cc) ICloneable.Clone**

[C#] object ICloneable.Clone();  
[C++] Object\* ICloneable::Clone();  
[VB] Function Clone() As Object Implements ICloneable.Clone  
[JScript] function ICloneable.Clone() : Object;  
MdiClient.ControlCollection class (System.Windows.Forms)

**a) ToString**

*Description*

Collection of controls...

**b) MdiClient.ControlCollection**

*Example Syntax:*

**c) ToString**

[C#] public MdiClient.ControlCollection(MdiClient owner);  
[C++] public: ControlCollection(MdiClient\* owner);  
[VB] Public Sub New(ByVal owner As MdiClient)  
[JScript] public function MdiClient.ControlCollection(owner : MdiClient);

*Description*

1        **d)     Count**

2        **e)     IsReadOnly**

3        **f)     Item**

4        **g)     Add**

6 [C#]        public        override        void        Add(Control        value);

7 [C++]        public:        void        Add(Control\*        value);

8 [VB]    Overrides    Public    Sub    Add(ByVal    value    As    Control)

9 [JScript]    public        override        function    Add(value        :    Control);

11 *Description*

12 Adds a control to the MDI Container. This child must be a Form that is marked as  
13 an MDI Child to be added to the container. You should not call this directly, but  
14 rather set the child form's (ctl) MDIParent property: // wrong Form child = new  
15 ChildForm(); this.getMdiClient().add(child); // right Form child = new  
16 ChildForm(); child.setMdiParent(this); Adds a control to the MDI Container. This  
17 child must be a Form that is marked as an MDI Child to be added to the  
18 container. You should not call this directly, but rather set the child form's (ctl)  
19 MDIParent property: // wrong Form child = new ChildForm();  
20 this.getMdiClient().add(child); // right Form child = new ChildForm();  
21 child.setMdiParent(this); MdiChild Form to add.

18        **h)     Remove**

20 [C#]        public        override        void        Remove(Control        value);

21 [C++]        public:        void        Remove(Control\*        value);

22 [VB]    Overrides    Public    Sub    Remove(ByVal    value    As    Control)

23 [JScript]    public        override        function    Remove(value        :    Control);

25 *Description*

Removes a child control. MDIChild Form to remove.

Form.ControlCollection class (System.Windows.Forms)

a) *ToString*

### *Description*

Represents a collection of controls on the form.

This class represents the collection of controls contained within a form. You can use the

**System.Windows.Forms.Form.ControlCollection.Add(System.Windows.Forms.Control)** method to add a control to the form and the

**System.Windows.Forms.Form.ControlCollection.Remove(System.Windows.Forms.Control)** method to remove the method from the form. The control collection represented by this class cannot be created without binding it to a specific form. As a result, you cannot create multiple instances of this control collection and interchange them with an active form to provide different control layouts.

b) *Form.ControlCollection*

*Example Syntax:*

c) *ToString*

[C#]            public            Form.ControlCollection(Form            owner);

[C++]            public:            ControlCollection(Form\*            owner);

[VB]            Public            Sub            New(ByVal            owner            As            Form)

[JScript]            public            function            Form.ControlCollection(owner            :            Form);

### *Description*

Initializes a new instance of the

**System.Windows.Forms.Form.ControlCollection** class.

This constructor allows you to properly bind the control collection to the form to enable controls to be added to the form. The **System.Windows.Forms.Form** to contain the controls added to the control collection.

*d) Count*

*e) IsReadOnly*

*f) Item*

*g) Add*

[C#] public override void Add(Control value);

[C++] public: void Add(Control\* value);

[VB] Overrides Public Sub Add(ByVal value As Control)

[JScript] public override function Add(value : Control);

### *Description*

Adds a control to the form.

You can use this method to adds controls to the form. If you want to add a group of already created controls to the form, use the **System.Windows.Forms.Control.ControlCollection.AddRange(System.Windows.Forms.Control[])** method of the **System.Windows.Forms.Control.ControlCollection** class. The **System.Windows.Forms.Control** to add to the form.

*h) Remove*

[C#] public override void Remove(Control value);

[C++] public: void Remove(Control\* value);

[VB] Overrides Public Sub Remove(ByVal value As Control)

[JScript] public override function Remove(value : Control);



## Description

Removes a control from the form.

You can use this method to remove controls that you no longer need in your form. If you want to display a control after it is removed, you will need to add the control back to the form using the **System.Windows.Forms.Form.ControlCollection.Add(System.Windows.Forms.Control)** method. To have a control remain on the form but not displayed, use the **System.Windows.Forms.Control.Visible** property of the control. A **System.Windows.Forms.Control** to remove from the form.

ControlEventArgs class (System.Windows.Forms)

### a) ToString

## Description

Provides data for the **System.Windows.Forms.Control.ControlAdded** and **System.Windows.Forms.Control.ControlRemoved** events.

For more information about handling events, see .

### b) ControlEventArgs

*Example Syntax:*

### c) ToString

|           |         |                                   |             |
|-----------|---------|-----------------------------------|-------------|
| [C#]      | public  | ControlEventArgs(Control          | control);   |
| [C++]     | public: | ControlEventArgs(Control*         | control);   |
| [VB]      | Public  | Sub New(ByVal control             | As Control) |
| [JScript] | public  | function ControlEventArgs(control | : Control); |

## Description

1 Initializes a new instance of the **System.Windows.Forms.ControlEventArgs**  
2 class for the specified control. The **System.Windows.Forms.Control** to store  
3 in this event.

4 *d) Control*

5 *e) ToString*

6 [C#] public Control Control {get;}

7 [C++] public: \_\_property Control\* get\_Control();

8 [VB] Public ReadOnly Property Control As Control

9 [JScript] public function get Control() : Control;

### 10 Description

11 Gets the control object used by this event.

12 ControlEventHandler delegate (System.Windows.Forms)

13 *a) ToString*

### 14 Description

15 Represents the method that will handle the  
16 **System.Windows.Forms.Control.ControlAdded** and  
17 **System.Windows.Forms.Control.ControlRemoved** events of the  
18 **System.Windows.Forms.Control** class. The source of the event. A  
19 **System.Windows.Forms.ControlEventArgs** that contains the event data.

20 When you create a **System.Windows.Forms.ControlEventArgs** delegate,  
21 you identify the method that will handle the event. To associate the event with  
22 your event handler, add an instance of the delegate to the event. The event  
23 handler is called whenever the event occurs, unless you remove the delegate.  
24 For more information about event handler delegates, see .

ControlPaint class (System.Windows.Forms)

a) *ToString*

### Description

Provides methods used to paint common Windows controls and their elements.

The methods contained in the **System.Windows.Forms.ControlPaint** class allow you to draw your own controls or elements of controls. You may control the drawing of your own controls if the **System.Windows.Forms.ControlStyles.UserPaint** bit is set to **true** for the control. You can get or set the style bits by calling the **System.Windows.Forms.Control.GetStyle(System.Windows.Forms.ControlStyles)** or **System.Windows.Forms.Control.SetStyle(System.Windows.Forms.ControlStyles, System.Boolean)** methods. You can set multiple style bits for any control. The **System.Windows.Forms.ControlStyles** enumeration members can be combined with bitwise operations.

b) *ContrastControlDark*

c) *ToString*

```
[C#]      public      static      Color      ContrastControlDark      {get;}
[C++]     public:     __property static Color get_ContrastControlDark();
[VB]      Public Shared ReadOnly Property ContrastControlDark As Color
[JScript] public static function get ContrastControlDark() : Color;
```

### Description

Gets the color to use as the **System.Drawing.SystemColors.ControlDark** color.

If the user has enabled the **System.Windows.Forms.SystemInformation.HighContrast** mode, this

property is set to **System.Drawing.SystemColors.WindowFrame** ;  
otherwise, it is set to **System.Drawing.SystemColors.ControlDark** .

*d) CreateHBitmap16Bit*

[C#] public static IntPtr CreateHBitmap16Bit(Bitmap bitmap, Color background);

[C++] public: static IntPtr CreateHBitmap16Bit(Bitmap\* bitmap, Color  
background);

[VB] Public Shared Function CreateHBitmap16Bit(ByVal bitmap As Bitmap,  
ByVal background As Color) As IntPtr

[JScript] public static function CreateHBitmap16Bit(bitmap : Bitmap, background  
: Color) : IntPtr;

*Description*

*e) CreateHBitmapColorMask*

[C#] public static IntPtr CreateHBitmapColorMask(Bitmap bitmap, IntPtr  
monochromeMask);

[C++] public: static IntPtr CreateHBitmapColorMask(Bitmap\* bitmap, IntPtr  
monochromeMask);

[VB] Public Shared Function CreateHBitmapColorMask(ByVal bitmap As  
Bitmap, ByVal monochromeMask As IntPtr) As IntPtr

[JScript] public static function CreateHBitmapColorMask(bitmap : Bitmap,  
monochromeMask : IntPtr) : IntPtr;

## Description

Creates a Win32 HBITMAP out of the image. You are responsible for de-allocating the HBITMAP with Windows.DeleteObject(handle). If the image uses transparency, the background will be filled with the specified color.

### f) *CreateHBitmapTransparencyMask*

[C#] public static IntPtr CreateHBitmapTransparencyMask(Bitmap bitmap);

[C++] public: static IntPtr CreateHBitmapTransparencyMask(Bitmap\* bitmap);

[VB] Public Shared Function CreateHBitmapTransparencyMask(ByVal bitmap As  
Bitmap) As IntPtr

[JScript] public static function CreateHBitmapTransparencyMask(bitmap :  
Bitmap) : IntPtr;

## Description

Creates a color mask for the specified bitmap that indicates which color should be displayed as transparent.

*Return Value:* The handle to the **System.Drawing.Bitmap** mask.

You are responsible for de-allocating the bitmap with Windows.DeleteObject(handle). The **System.Drawing.Bitmap** to create the transparency mask for.

### g) *Dark*

[C#] public static Color Dark(Color baseColor);

[C++] public: static Color Dark(Color baseColor);

[VB] Public Shared Function Dark(ByVal baseColor As Color) As Color

[JScript] public static function Dark(baseColor : Color) : Color;

## Description

Creates a new dark color object for the control from the specified color.

*Return Value:* A **System.Drawing.Color** that represents the dark color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to a **System.Drawing.SystemColors.ControlDark** color; otherwise, the color's luminosity value is decreased. The **System.Drawing.Color** to be darkened.

### *h) Dark*

```
[C#] public static Color Dark(Color baseColor, float percOfDarkDark);
```

```
[C++] public: static Color Dark(Color baseColor, float percOfDarkDark);
```

```
[VB] Public Shared Function Dark(ByVal baseColor As Color, ByVal  
percOfDarkDark As Single) As Color
```

```
[JScript] public static function Dark(baseColor : Color, percOfDarkDark : float) :  
Color; Creates a new dark color object for the control.
```

## Description

Creates a new dark color object for the control from the specified color and darkens it by the specified percentage.

*Return Value:* A **System.Drawing.Color** that represent the dark color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to a **System.Drawing.SystemColors.ControlDark** color; otherwise, the color's luminosity value is decreased. The **System.Drawing.Color** to be darkened. The percentage to darken the specified **System.Drawing.Color**.

i) **DarkDark**

```
[C#]      public      static      Color      DarkDark(Color      baseColor);  
[C++]     public:     static      Color      DarkDark(Color      baseColor);  
[VB]      Public Shared Function DarkDark(ByVal baseColor As Color) As Color  
[JScript] public static function DarkDark(baseColor : Color) : Color;
```

*Description*

Creates a new dark color object for the control from the specified color.

*Return Value:* A **System.Drawing.Color** that represents the dark color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to the **System.Drawing.SystemColors.ControlDarkDark** color; otherwise, the color's luminosity value is increased. The **System.Drawing.Color** to be darkened.

j) **DrawBorder**

```
[C#] public static void DrawBorder(Graphics graphics, Rectangle bounds, Color  
color, ButtonBorderStyle style);  
[C++] public: static void DrawBorder(Graphics* graphics, Rectangle bounds,  
Color color, ButtonBorderStyle style);  
[VB] Public Shared Sub DrawBorder(ByVal graphics As Graphics, ByVal bounds  
As Rectangle, ByVal color As Color, ByVal style As ButtonBorderStyle)  
[JScript] public static function DrawBorder(graphics : Graphics, bounds :  
Rectangle, color : Color, style : ButtonBorderStyle); Draws a border on a button-  
style control.
```

## Description

Draws a border with the specified style and color, on the specified graphics surface, and within the specified bounds on a button-style control. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the border. The **System.Drawing.Color** of the border. The **System.Windows.Forms.ButtonBorderStyle** of the border.

### k) DrawBorder

```
[C#] public static void DrawBorder(Graphics graphics, Rectangle bounds, Color leftColor, int leftWidth, ButtonBorderStyle leftStyle, Color topColor, int topWidth, ButtonBorderStyle topStyle, Color rightColor, int rightWidth, ButtonBorderStyle rightStyle, Color bottomColor, int bottomWidth, ButtonBorderStyle bottomStyle);
```

```
[C++] public: static void DrawBorder(Graphics* graphics, Rectangle bounds, Color leftColor, int leftWidth, ButtonBorderStyle leftStyle, Color topColor, int topWidth, ButtonBorderStyle topStyle, Color rightColor, int rightWidth, ButtonBorderStyle rightStyle, Color bottomColor, int bottomWidth, ButtonBorderStyle bottomStyle);
```

```
[VB] Public Shared Sub DrawBorder(ByVal graphics As Graphics, ByVal bounds As Rectangle, ByVal leftColor As Color, ByVal leftWidth As Integer, ByVal leftStyle As ButtonBorderStyle, ByVal topColor As Color, ByVal topWidth As Integer, ByVal topStyle As ButtonBorderStyle, ByVal rightColor As Color, ByVal rightWidth As Integer, ByVal rightStyle As ButtonBorderStyle, ByVal bottomColor As Color, ByVal bottomWidth As Integer, ByVal bottomStyle As ButtonBorderStyle)
```



```

1 [JScript] public static function DrawBorder(graphics : Graphics, bounds :
2 Rectangle, leftColor : Color, leftWidth : int, leftStyle : ButtonBorderStyle,
3 topColor : Color, topWidth : int, topStyle : ButtonBorderStyle, rightColor : Color,
4 rightWidth : int, rightStyle : ButtonBorderStyle, bottomColor : Color,
5 bottomWidth : int, bottomStyle : ButtonBorderStyle);

```

### 7 *Description*

8 Draws a border on a button-style control with the specified styles, colors, and  
9 border widths, on the specified graphics surface, and within the specified  
10 bounds. The **System.Drawing.Graphics** object to draw on. The  
11 **System.Drawing.Rectangle** that represents the dimensions of the border. The  
12 **System.Drawing.Color** of the left of the border. The  
13 **System.Drawing.Pen.Width** of the left border. The  
14 **System.Windows.Forms.ButtonBorderStyle** of the left border. The  
15 **System.Drawing.Color** of the top of the border. The  
16 **System.Drawing.Pen.Width** of the top border. The  
17 **System.Windows.Forms.ButtonBorderStyle** of the top border. The  
18 **System.Drawing.Color** of the right of the border. The  
19 **System.Drawing.Pen.Width** of the right border. The  
20 **System.Windows.Forms.ButtonBorderStyle** of the right border. The  
21 **System.Drawing.Color** of the bottom of the border. The  
22 **System.Drawing.Pen.Width** of the left border. The  
23 **System.Windows.Forms.ButtonBorderStyle** of the bottom border.

### 17 *1) DrawBorder3D*

```

19 [C#] public static void DrawBorder3D(Graphics graphics, Rectangle rectangle);
20 [C++] public: static void DrawBorder3D(Graphics* graphics, Rectangle
21 rectangle);
22 [VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal
23 rectangle As Rectangle)
24 [JScript] public static function DrawBorder3D(graphics : Graphics, rectangle :
25

```

Rectangle); Draws a three-dimensional style border on a control.

### *Description*

Draws a three-dimensional style border on the specified graphics surface and within the specified bounds on a control.

The **System.Windows.Forms.Border3DStyle.Etched** style is used by default to draw the border. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the border.

#### *m) DrawBorder3D*

[C#] public static void DrawBorder3D(Graphics graphics, Rectangle rectangle, Border3DStyle style);

[C++] public: static void DrawBorder3D(Graphics\* graphics, Rectangle rectangle, Border3DStyle style);

[VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal rectangle As Rectangle, ByVal style As Border3DStyle)

[JScript] public static function DrawBorder3D(graphics : Graphics, rectangle : Rectangle, style : Border3DStyle);

### *Description*

Draws a three-dimensional style border with the specified style, on the specified graphics surface, and within the specified bounds on a control. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the border. The **System.Windows.Forms.Border3DStyle** to use when drawing the border.

#### *n) DrawBorder3D*

[C#] public static void DrawBorder3D(Graphics graphics, Rectangle rectangle,

```

1 Border3DStyle          style,          Border3DSide          sides);
2 [C++] public: static void DrawBorder3D(Graphics* graphics, Rectangle rectangle,
3 Border3DStyle          style,          Border3DSide          sides);
4 [VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal
5 rectangle As Rectangle, ByVal style As Border3DStyle, ByVal sides As
6 Border3DSide)
7 [JScript] public static function DrawBorder3D(graphics : Graphics, rectangle :
8 Rectangle, style : Border3DStyle, sides : Border3DSide);
9

```

### *Description*

Draws a three-dimensional style border with the specified style, on the specified graphics surface and sides, and within the specified bounds on a control. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the border. The **System.Windows.Forms.Border3DStyle** to use when drawing the border. The **System.Windows.Forms.Border3DSide** of the rectangle to draw the border on.

#### *o) DrawBorder3D*

```

17 [C#] public static void DrawBorder3D(Graphics graphics, int x, int y, int width,
18 int height);
19 [C++] public: static void DrawBorder3D(Graphics* graphics, int x, int y, int
20 width, int height);
21 [VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal x As
22 Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer)
23 [JScript] public static function DrawBorder3D(graphics : Graphics, x : int, y : int,
24 width : int, height : int);
25

```

## Description

Draws a three-dimensional style border on the specified graphics surface and within the specified bounds on a control.

The **System.Windows.Forms.Border3DStyle.Etched** style is used by default to draw the border. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Y** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Width** of the border rectangle. The **System.Drawing.Rectangle.Height** of the border rectangle.

### p) *DrawBorder3D*

```
[C#] public static void DrawBorder3D(Graphics graphics, int x, int y, int width,
int height, Border3DStyle style);
```

```
[C++] public: static void DrawBorder3D(Graphics* graphics, int x, int y, int
width, int height, Border3DStyle style);
```

```
[VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal x As
Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
ByVal style As Border3DStyle)
```

```
[JScript] public static function DrawBorder3D(graphics : Graphics, x : int, y : int,
width : int, height : int, style : Border3DStyle);
```

## Description

Draws a three-dimensional style border with the specified style, on the specified graphics surface, and within the specified bounds on a control. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Y** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Width** of the border rectangle. The **System.Drawing.Rectangle.Height** of the border rectangle.

The **System.Windows.Forms.Border3DStyle** to use when drawing the border.

*q) DrawBorder3D*

```
[C#] public static void DrawBorder3D(Graphics graphics, int x, int y, int width,
int height, Border3DStyle style, Border3DSide sides);
[C++] public: static void DrawBorder3D(Graphics* graphics, int x, int y, int
width, int height, Border3DStyle style, Border3DSide sides);
[VB] Public Shared Sub DrawBorder3D(ByVal graphics As Graphics, ByVal x As
Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
ByVal style As Border3DStyle, ByVal sides As Border3DSide)
[JavaScript] public static function DrawBorder3D(graphics : Graphics, x : int, y : int,
width : int, height : int, style : Border3DStyle, sides : Border3DSide);
```

*Description*

Draws a three-dimensional style border with the specified style, on the specified graphics surface and side, and within the specified bounds on a control. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Y** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Width** of the border rectangle. The **System.Drawing.Rectangle.Height** of the border rectangle. The **System.Windows.Forms.Border3DStyle** to use when drawing the border. The **System.Windows.Forms.Border3DSide** of the rectangle to draw the border on.

*r) DrawButton*

```
[C#] public static void DrawButton(Graphics graphics, Rectangle rectangle,
ButtonState state);
```

```

1 [C++] public: static void DrawButton(Graphics* graphics, Rectangle rectangle,
2 ButtonState state);
3 [VB] Public Shared Sub DrawButton(ByVal graphics As Graphics, ByVal
4 rectangle As Rectangle, ByVal state As ButtonState)
5 [JScript] public static function DrawButton(graphics : Graphics, rectangle :
6 Rectangle, state : ButtonState); Draws a button control.

```

### *Description*

Draws a button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the button. The **System.Windows.Forms.ButtonState** to draw the button in.

#### *s) DrawButton*

```

13 [C#] public static void DrawButton(Graphics graphics, int x, int y, int width, int
14 height, ButtonState state);
15 [C++] public: static void DrawButton(Graphics* graphics, int x, int y, int width,
16 int height, ButtonState state);
17 [VB] Public Shared Sub DrawButton(ByVal graphics As Graphics, ByVal x As
18 Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
19 ByVal state As ButtonState)
20 [JScript] public static function DrawButton(graphics : Graphics, x : int, y : int,
21 width : int, height : int, state : ButtonState);
22

```

### *Description*

Draws a button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to

draw on. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Width** of the button. The **System.Drawing.Rectangle.Height** of the button. The **System.Windows.Forms.ButtonState** to draw the button in.

*t) DrawCaptionButton*

```
[C#] public static void DrawCaptionButton(Graphics graphics, Rectangle  
rectangle, CaptionButton button, ButtonState state);
```

```
[C++] public: static void DrawCaptionButton(Graphics* graphics, Rectangle  
rectangle, CaptionButton button, ButtonState state);
```

```
[VB] Public Shared Sub DrawCaptionButton(ByVal graphics As Graphics, ByVal  
rectangle As Rectangle, ByVal button As CaptionButton, ByVal state As  
ButtonState)
```

```
[JScript] public static function DrawCaptionButton(graphics : Graphics, rectangle  
: Rectangle, button : CaptionButton, state : ButtonState); Draws a caption button  
control.
```

*Description*

Draws the specified caption button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the caption button. The **System.Windows.Forms.CaptionButton** to draw. The **System.Windows.Forms.ButtonState** to draw the caption button in.

*u) DrawCaptionButton*

```
[C#] public static void DrawCaptionButton(Graphics graphics, int x, int y, int
```

```

1 width, int height, CaptionButton button, ButtonState state);
2 [C++] public: static void DrawCaptionButton(Graphics* graphics, int x, int y, int
3 width, int height, CaptionButton button, ButtonState state);
4 [VB] Public Shared Sub DrawCaptionButton(ByVal graphics As Graphics, ByVal
5 x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As
6 Integer, ByVal button As CaptionButton, ByVal state As ButtonState)
7 [JScript] public static function DrawCaptionButton(graphics : Graphics, x : int, y :
8 int, width : int, height : int, button : CaptionButton, state : ButtonState);

```

### *Description*

Draws the specified caption button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the top left of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the top left of the drawing rectangle. The **System.Drawing.Rectangle.Width** of the drawing rectangle. The **System.Drawing.Rectangle.Height** of the drawing rectangle. The **System.Windows.Forms.CaptionButton** to draw. The **System.Windows.Forms.ButtonState** to draw the caption button in.

### *v) DrawCheckBox*

```

18 [C#] public static void DrawCheckBox(Graphics graphics, Rectangle rectangle,
19 ButtonState state);
20 [C++] public: static void DrawCheckBox(Graphics* graphics, Rectangle
21 rectangle, ButtonState state);
22 [VB] Public Shared Sub DrawCheckBox(ByVal graphics As Graphics, ByVal
23 rectangle As Rectangle, ByVal state As ButtonState)
24 [JScript] public static function DrawCheckBox(graphics : Graphics, rectangle :
25

```



Rectangle, state : ButtonState); Draws a check box control.

### Description

Draws a check box control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the check box. The **System.Windows.Forms.ButtonState** to draw the check box in.

#### w) DrawCheckBox

[C#] public static void DrawCheckBox(Graphics graphics, int x, int y, int width, int height, ButtonState state);

[C++] public: static void DrawCheckBox(Graphics\* graphics, int x, int y, int width, int height, ButtonState state);

[VB] Public Shared Sub DrawCheckBox(ByVal graphics As Graphics, ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal state As ButtonState)

[JScript] public static function DrawCheckBox(graphics : Graphics, x : int, y : int, width : int, height : int, state : ButtonState);

### Description

Draws a check box control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Width** of the check box. The **System.Drawing.Rectangle.Height** of the check box. The **System.Windows.Forms.ButtonState** to draw the check box in.

*x) DrawComboButton*

```
[C#] public static void DrawComboButton(Graphics graphics, Rectangle
rectangle, ButtonState state);
[C++] public: static void DrawComboButton(Graphics* graphics, Rectangle
rectangle, ButtonState state);
[VB] Public Shared Sub DrawComboButton(ByVal graphics As Graphics, ByVal
rectangle As Rectangle, ByVal state As ButtonState)
[JavaScript] public static function DrawComboButton(graphics : Graphics, rectangle :
Rectangle, state : ButtonState); Draws a drop-down button on a combo box
control.
```

*Description*

Draws a drop-down button on a combo box control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the combo box. The **System.Windows.Forms.ButtonState** to draw the combo box in.

*y) DrawComboButton*

```
[C#] public static void DrawComboButton(Graphics graphics, int x, int y, int
width, int height, ButtonState state);
[C++] public: static void DrawComboButton(Graphics* graphics, int x, int y, int
width, int height, ButtonState state);
[VB] Public Shared Sub DrawComboButton(ByVal graphics As Graphics, ByVal
x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As
Integer, ByVal state As ButtonState)
```

```

1 [JScript] public static function DrawComboButton(graphics : Graphics, x : int, y :
2 int, width : int, height : int, state : ButtonState);
3

```

#### 4 *Description*

5 Draws a drop-down button on a combo box control in the specified state, on the specified graphics surface, and within the specified bounds. The

6 **System.Drawing.Graphics** object to draw on. The

7 **System.Drawing.Rectangle.X** coordinate of the top left of the border rectangle. The **System.Drawing.Rectangle.Y** coordinate of the top left of the border rectangle. The **System.Windows.Forms.Control.Width** of the combo box. The **System.Windows.Forms.Control.Height** of the combo box. The **System.Windows.Forms.ButtonState** to draw the combo box in.

#### 10 z) *DrawContainerGrabHandle*

```

12 [C#] public static void DrawContainerGrabHandle(Graphics graphics, Rectangle
13 bounds);
14

```

```

15 [C++] public: static void DrawContainerGrabHandle(Graphics* graphics,
16 Rectangle bounds);
17

```

```

18 [VB] Public Shared Sub DrawContainerGrabHandle(ByVal graphics As Graphics,
19 ByVal bounds As Rectangle)
20

```

```

21 [JScript] public static function DrawContainerGrabHandle(graphics : Graphics,
22 bounds : Rectangle);
23

```

#### 21 *Description*

22 Draws a container control grab handle glyph on the specified graphics surface and within the specified bounds.

23 Grab handles are used by containers to indicate to the user that the user can directly manipulate the containers. The manipulation can consist of actions such as sizing and moving. The **System.Drawing.Graphics** object to draw on. The

**System.Drawing.Rectangle** that represents the dimensions of the grab handle glyph.

*aa) DrawFocusRectangle*

[C#] public static void DrawFocusRectangle(Graphics graphics, Rectangle rectangle);

[C++] public: static void DrawFocusRectangle(Graphics\* graphics, Rectangle rectangle);

[VB] Public Shared Sub DrawFocusRectangle(ByVal graphics As Graphics, ByVal rectangle As Rectangle)

[JScript] public static function DrawFocusRectangle(graphics : Graphics, rectangle : Rectangle); Draws a focus rectangle.

*Description*

Draws a focus rectangle on the specified graphics surface and within the specified bounds.

A focus rectangle is a dotted rectangle that Windows uses to indicate what control has the current keyboard focus. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the grab handle glyph.

*bb) DrawFocusRectangle*

[C#] public static void DrawFocusRectangle(Graphics graphics, Rectangle rectangle, Color foreColor, Color backColor);

[C++] public: static void DrawFocusRectangle(Graphics\* graphics, Rectangle rectangle, Color foreColor, Color backColor);

[VB] Public Shared Sub DrawFocusRectangle(ByVal graphics As Graphics,

ByVal rectangle As Rectangle, ByVal foreColor As Color, ByVal backColor As Color)

[JScript] public static function DrawFocusRectangle(graphics : Graphics, rectangle : Rectangle, foreColor : Color, backColor : Color);

#### *Description*

Draws a focus rectangle on the specified graphics surface and within the specified bounds.

A focus rectangle is a dotted rectangle that Windows uses to indicate what control has the current keyboard focus. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the grab handle glyph. The **System.Windows.Forms.Control.ForeColor** of the object to draw the focus rectangle on. The **System.Windows.Forms.Control.BackColor** of the object to draw the focus rectangle on.

#### *cc) DrawGrabHandle*

[C#] public static void DrawGrabHandle(Graphics graphics, Rectangle rectangle, bool primary, bool enabled);

[C++] public: static void DrawGrabHandle(Graphics\* graphics, Rectangle rectangle, bool primary, bool enabled);

[VB] Public Shared Sub DrawGrabHandle(ByVal graphics As Graphics, ByVal rectangle As Rectangle, ByVal primary As Boolean, ByVal enabled As Boolean)

[JScript] public static function DrawGrabHandle(graphics : Graphics, rectangle : Rectangle, primary : Boolean, enabled : Boolean);

#### *Description*

Draws a standard selection grab handle glyph on the specified graphics surface, within the specified bounds, and in the specified state and style.

Grab handles are used by objects to indicate to the user that the user can directly manipulate the object. The manipulation can consist of actions such as sizing and moving. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the grab handle glyph. **true** to draw the handle as a primary grab handle; otherwise, **false**. **true** to draw the handle in an enabled state; otherwise, **false**.

*dd) DrawGrid*

```
[C#] public static void DrawGrid(Graphics graphics, Rectangle area, Size
pixelsBetweenDots, Color backColor);
```

```
[C++] public: static void DrawGrid(Graphics* graphics, Rectangle area, Size
pixelsBetweenDots, Color backColor);
```

```
[VB] Public Shared Sub DrawGrid(ByVal graphics As Graphics, ByVal area As
Rectangle, ByVal pixelsBetweenDots As Size, ByVal backColor As Color)
```

```
[JScript] public static function DrawGrid(graphics : Graphics, area : Rectangle,
pixelsBetweenDots : Size, backColor : Color);
```

*Description*

Draws a grid of one-pixel dots with the specified spacing, within the specified bounds, on the specified graphics surface, and in the specified color.

The *backColor* parameter is used to calculate the fill color of the dots so that the grid is always visible against the background. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the grab handle glyph. The **System.Drawing.Size.Height** and **System.Drawing.Size.Width** between the dots of the grid. The **System.Drawing.Color** of the background behind the grid.

*ee) DrawImageDisabled*

```
[C#] public static void DrawImageDisabled(Graphics graphics, Image image, int
x, int y, Color background);
```

```

1 [C++] public: static void DrawImageDisabled(Graphics* graphics, Image* image,
2 int x, int y, Color background);
3 [VB] Public Shared Sub DrawImageDisabled(ByVal graphics As Graphics, ByVal
4 image As Image, ByVal x As Integer, ByVal y As Integer, ByVal background As
5 Color)
6 [JScript] public static function DrawImageDisabled(graphics : Graphics, image :
7 Image, x : int, y : int, background : Color);
8

```

### *Description*

Draws the specified image in a disabled state.

The *background* parameter is used to calculate the fill color of the disabled image so that it is always visible against the background. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Image** to draw. The **System.Drawing.Rectangle.X** coordinate of the top left of the border image. The **System.Drawing.Rectangle.Y** coordinate of the top left of the border image. The **System.Drawing.Color** of the background behind the image.

### *ff) DrawLockedFrame*

```

17 [C#] public static void DrawLockedFrame(Graphics graphics, Rectangle
18 rectangle, bool primary);
19 [C++] public: static void DrawLockedFrame(Graphics* graphics, Rectangle
20 rectangle, bool primary);
21 [VB] Public Shared Sub DrawLockedFrame(ByVal graphics As Graphics, ByVal
22 rectangle As Rectangle, ByVal primary As Boolean)
23 [JScript] public static function DrawLockedFrame(graphics : Graphics, rectangle :
24 Rectangle, primary : Boolean);
25

```

## Description

Draws a locked selection frame on the screen within the specified bounds and on the specified graphics surface. Specifies whether to draw the frame with the primary selected colors. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the frame. **true** to draw this frame with the colors used for the primary selection; otherwise, **false**.

### gg) DrawMenuGlyph

```
[C#] public static void DrawMenuGlyph(Graphics graphics, Rectangle rectangle,
MenuGlyph glyph);
[C++] public: static void DrawMenuGlyph(Graphics* graphics, Rectangle
rectangle, MenuGlyph glyph);
[VB] Public Shared Sub DrawMenuGlyph(ByVal graphics As Graphics, ByVal
rectangle As Rectangle, ByVal glyph As MenuGlyph)
[JScript] public static function DrawMenuGlyph(graphics : Graphics, rectangle :
Rectangle, glyph : MenuGlyph); Draws a menu glyph on a menu item control.
```

## Description

Draws the specified menu glyph on a menu item control within the specified bounds and on the specified surface.

When owner-drawing **System.Windows.Forms.MenuItem** controls, you need to verify property values to determine the correct glyph to draw or remove. For example, when the **System.Windows.Forms.MenuItem.Checked** property is set to **true**, you need to draw a **System.Windows.Forms.MenuGlyph.Checkmark** glyph on the **System.Windows.Forms.MenuItem**. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the glyph. The **System.Windows.Forms.MenuGlyph** to draw.



## hh) DrawMenuGlyph

```
[C#] public static void DrawMenuGlyph(Graphics graphics, int x, int y, int width,
int height, MenuGlyph glyph);
[C++] public: static void DrawMenuGlyph(Graphics* graphics, int x, int y, int
width, int height, MenuGlyph glyph);
[VB] Public Shared Sub DrawMenuGlyph(ByVal graphics As Graphics, ByVal x
As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
ByVal glyph As MenuGlyph)
[JScript] public static function DrawMenuGlyph(graphics : Graphics, x : int, y :
int, width : int, height : int, glyph : MenuGlyph);
```

### Description

Draws the specified menu glyph on a menu item control with the specified bounds and on the specified surface.

When owner-drawing **System.Windows.Forms.MenuItem** controls, you need to verify property values to determine the correct glyph to draw or remove. For example, when the **System.Windows.Forms.MenuItem.Checked** property is set to **true**, you need to draw a **System.Windows.Forms.MenuGlyph.Checkmark** glyph on the **System.Windows.Forms.MenuItem**. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Width** of the menu glyph. The **System.Drawing.Rectangle.Height** of the menu glyph. The **System.Windows.Forms.MenuGlyph** to draw.

## ii) DrawMixedCheckBox

```
[C#] public static void DrawMixedCheckBox(Graphics graphics, Rectangle
```

```

1 rectangle, ButtonState state);
2 [C++] public: static void DrawMixedCheckBox(Graphics* graphics, Rectangle
3 rectangle, ButtonState state);
4 [VB] Public Shared Sub DrawMixedCheckBox(ByVal graphics As Graphics,
5 ByVal rectangle As Rectangle, ByVal state As ButtonState)
6 [JScript] public static function DrawMixedCheckBox(graphics : Graphics,
7 rectangle : Rectangle, state : ButtonState); Draws a three-state check box control.
8

```

### 9 *Description*

10 Draws a three-state check box control in the specified state, on the specified  
11 graphics surface, and within the specified bounds. The  
12 **System.Drawing.Graphics** object to draw on. The  
13 **System.Drawing.Rectangle** that represents the dimensions of the check box.  
14 The **System.Windows.Forms.ButtonState** to draw the check box in.

### 13 *jj) DrawMixedCheckBox*

```

14
15 [C#] public static void DrawMixedCheckBox(Graphics graphics, int x, int y, int
16 width, int height, ButtonState state);
17 [C++] public: static void DrawMixedCheckBox(Graphics* graphics, int x, int y,
18 int width, int height, ButtonState state);
19 [VB] Public Shared Sub DrawMixedCheckBox(ByVal graphics As Graphics,
20 ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height
21 As Integer, ByVal state As ButtonState)
22 [JScript] public static function DrawMixedCheckBox(graphics : Graphics, x : int,
23 y : int, width : int, height : int, state : ButtonState);
24
25

```

## Description

Draws a three-state check box control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Width** of the check box. The **System.Drawing.Rectangle.Height** of the check box. The **System.Windows.Forms.ButtonState** to draw the check box in.

### kk) DrawRadioButton

[C#] public static void DrawRadioButton(Graphics graphics, Rectangle rectangle, ButtonState state);

[C++] public: static void DrawRadioButton(Graphics\* graphics, Rectangle rectangle, ButtonState state);

[VB] Public Shared Sub DrawRadioButton(ByVal graphics As Graphics, ByVal rectangle As Rectangle, ByVal state As ButtonState)

[JScript] public static function DrawRadioButton(graphics : Graphics, rectangle : Rectangle, state : ButtonState); Draws a radio button control.

## Description

Draws a radio button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the radio button. The **System.Windows.Forms.ButtonState** to draw the radio button in.

## II) *DrawRadioButton*

```
[C#] public static void DrawRadioButton(Graphics graphics, int x, int y, int width,
int height, ButtonState state);
[C++] public: static void DrawRadioButton(Graphics* graphics, int x, int y, int
width, int height, ButtonState state);
[VB] Public Shared Sub DrawRadioButton(ByVal graphics As Graphics, ByVal x
As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
ByVal state As ButtonState)
[JScript] public static function DrawRadioButton(graphics : Graphics, x : int, y :
int, width : int, height : int, state : ButtonState);
```

### *Description*

Draws a radio button control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the drawing rectangle. The **System.Windows.Forms.Control.Width** of the radio button. The **System.Windows.Forms.Control.Height** of the radio button. The **System.Windows.Forms.ButtonState** to draw the radio button in.

## mm) *DrawReversibleFrame*

```
[C#] public static void DrawReversibleFrame(Rectangle rectangle, Color
BackColor, FrameStyle style);
[C++] public: static void DrawReversibleFrame(Rectangle rectangle, Color
BackColor, FrameStyle style);
[VB] Public Shared Sub DrawReversibleFrame(ByVal rectangle As Rectangle,
```

```

1 ByVal backColor As Color, ByVal style As FrameStyle)
2 [JScript] public static function DrawReversibleFrame(rectangle : Rectangle,
3 backColor : Color, style : FrameStyle);

```

#### *Description*

Draws a reversible frame on the screen within the specified bounds, with the specified background color, and in the specified state.

The *backColor* parameter is used to calculate the fill color of the frame so that it is always visible against the background. The **System.Drawing.Rectangle** that represents the dimensions of the rectangle to draw, in screen coordinates. The **System.Drawing.Color** of the background behind the frame. The **System.Windows.Forms.FrameStyle** of the frame.

#### *nn) DrawReversibleLine*

```

13 [C#] public static void DrawReversibleLine(Point start, Point end, Color
14 backColor);

```

```

15 [C++] public: static void DrawReversibleLine(Point start, Point end, Color
16 backColor);

```

```

17 [VB] Public Shared Sub DrawReversibleLine(ByVal start As Point, ByVal end As
18 Point, ByVal backColor As Color)

```

```

19 [JScript] public static function DrawReversibleLine(start : Point, end : Point,
20 backColor : Color);

```

#### *Description*

Draws a reversible line on the screen within the specified starting and ending points and with the specified background color.

The *backColor* parameter is used to calculate the fill color of the line so that it is always visible against the background. The starting **System.Drawing.Point** of the line, in screen coordinates. The ending **System.Drawing.Point** of the line,

in screen coordinates. The **System.Drawing.Color** of the background behind the line.

*oo) DrawScrollButton*

[C#] public static void DrawScrollButton(Graphics graphics, Rectangle rectangle, ScrollButton button, ButtonState state);

[C++] public: static void DrawScrollButton(Graphics\* graphics, Rectangle rectangle, ScrollButton button, ButtonState state);

[VB] Public Shared Sub DrawScrollButton(ByVal graphics As Graphics, ByVal rectangle As Rectangle, ByVal button As ScrollButton, ByVal state As ButtonState)

[JScript] public static function DrawScrollButton(graphics : Graphics, rectangle : Rectangle, button : ScrollButton, state : ButtonState); Draws a scroll button on a scroll bar control.

*Description*

Draws the specified scroll button on a scroll bar control in the specified state, on the specified graphics surface, and within the specified bounds. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Rectangle** that represents the dimensions of the glyph. The **System.Windows.Forms.ScrollButton** to draw. The **System.Windows.Forms.ButtonState** that represents the state to draw the scroll button in.

*pp) DrawScrollButton*

[C#] public static void DrawScrollButton(Graphics graphics, int x, int y, int width, int height, ScrollButton button, ButtonState state);

[C++] public: static void DrawScrollButton(Graphics\* graphics, int x, int y, int

```

1 width, int height, ScrollButton button, ButtonState state);
2 [VB] Public Shared Sub DrawScrollBar(ByVal graphics As Graphics, ByVal x
3 As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer,
4 ByVal button As ScrollButton, ByVal state As ButtonState)
5 [JScript] public static function DrawScrollBar(graphics : Graphics, x : int, y :
6 int, width : int, height : int, button : ScrollButton, state : ButtonState);
7

```

### 8 *Description*

9 Draws the specified scroll button on a scroll bar control in the specified state, on  
10 the specified graphics surface, and within the specified bounds. The  
11 **System.Drawing.Graphics** object to draw on. The  
12 **System.Drawing.Rectangle.X** coordinate of the upper left corner of the  
13 drawing rectangle. The **System.Drawing.Rectangle.Y** coordinate of the upper  
14 left corner of the drawing rectangle. The **System.Drawing.Rectangle.Width**  
15 of the scroll button. The **System.Drawing.Rectangle.Height** of the scroll  
16 button. The **System.Windows.Forms.ScrollButton** to draw. The  
17 **System.Windows.Forms.ButtonState** that represents the state to draw the  
18 scroll button in.

### 15 *qq) DrawSelectionFrame*

```

17 [C#] public static void DrawSelectionFrame(Graphics graphics, bool active,
18 Rectangle outsideRect, Rectangle insideRect, Color backColor);
19 [C++] public: static void DrawSelectionFrame(Graphics* graphics, bool active,
20 Rectangle outsideRect, Rectangle insideRect, Color backColor);
21 [VB] Public Shared Sub DrawSelectionFrame(ByVal graphics As Graphics,
22 ByVal active As Boolean, ByVal outsideRect As Rectangle, ByVal insideRect As
23 Rectangle, ByVal backColor As Color)
24 [JScript] public static function DrawSelectionFrame(graphics : Graphics, active :
25 Boolean, outsideRect : Rectangle, insideRect : Rectangle, backColor : Color);

```

## Description

Draws a standard selection frame in the specified state, on the specified graphics surface, with the specified inner and outer dimensions, and with the specified background color.

A selection frame is a frame that is drawn around a selected component at design-time. The **System.Drawing.Graphics** object to draw on. **true** to draw the selection frame in an active state; otherwise, **false**. The **System.Drawing.Rectangle** that represents the outer boundary of the selection frame. The **System.Drawing.Rectangle** that represents the inner boundary of the selection frame. The **System.Drawing.Color** of the background behind the frame.

### *rr) DrawSizeGrip*

```
[C#] public static void DrawSizeGrip(Graphics graphics, Color backColor,
Rectangle bounds);
```

```
[C++] public: static void DrawSizeGrip(Graphics* graphics, Color backColor,
Rectangle bounds);
```

```
[VB] Public Shared Sub DrawSizeGrip(ByVal graphics As Graphics, ByVal
backColor As Color, ByVal bounds As Rectangle)
```

```
[JScript] public static function DrawSizeGrip(graphics : Graphics, backColor :
Color, bounds : Rectangle); Draws a size grip on a form.
```

## Description

Draws a size grip on a form with the specified bounds and background color and on the specified graphics surface.

The *backColor* parameter is used to calculate the color of the size grip so that it is always visible against the background. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Color** of the background used to determine the colors of the size grip. The **System.Drawing.Rectangle** that represents the dimensions of the size grip.



ss) *DrawSizeGrip*

```
[C#] public static void DrawSizeGrip(Graphics graphics, Color backColor, int x,
int y, int width, int height);
[C++] public: static void DrawSizeGrip(Graphics* graphics, Color backColor, int
x, int y, int width, int height);
[VB] Public Shared Sub DrawSizeGrip(ByVal graphics As Graphics, ByVal
backColor As Color, ByVal x As Integer, ByVal y As Integer, ByVal width As
Integer, ByVal height As Integer)
[JScript] public static function DrawSizeGrip(graphics : Graphics, backColor :
Color, x : int, y : int, width : int, height : int);
```

*Description*

Draws a size grip on a form with the specified bounds and background color and on the specified graphics surface.

The *backColor* parameter is used to calculate the color of the size grip so that it is always visible against the background. The **System.Drawing.Graphics** object to draw on. The **System.Drawing.Color** of the background used to determine the colors of the size grip. The **System.Drawing.Rectangle.X** coordinate of the upper left corner of the size grip. The **System.Drawing.Rectangle.Y** coordinate of the upper left corner of the size grip. The **System.Drawing.Rectangle.Width** of the size grip. The **System.Drawing.Rectangle.Height** of the size grip.

tt) *DrawStringDisabled*

```
[C#] public static void DrawStringDisabled(Graphics graphics, string s, Font font,
Color color, RectangleF layoutRectangle, StringFormat format);
[C++] public: static void DrawStringDisabled(Graphics* graphics, String* s,
Font* font, Color color, RectangleF layoutRectangle, StringFormat* format);
```

```

1 [VB] Public Shared Sub DrawStringDisabled(ByVal graphics As Graphics, ByVal
2 s As String, ByVal font As Font, ByVal color As Color, ByVal layoutRectangle
3 As RectangleF, ByVal format As StringFormat)
4 [JScript] public static function DrawStringDisabled(graphics : Graphics, s : String,
5 font : Font, color : Color, layoutRectangle : RectangleF, format : StringFormat);
6

```

### *Description*

Draws the specified string in a disabled state on the specified graphics surface, within the specified bounds, and in the specified font, color, and format. The **System.Drawing.Graphics** object to draw on. The **System.String** to draw. The **System.Drawing.Font** to draw the string with. The **System.Drawing.Color** to draw the string with. The **System.Drawing.RectangleF** object that represents the dimensions of the string. The **System.Drawing.StringFormat** to apply to the string.

### *uu) FillReversibleRectangle*

```

14 [C#] public static void FillReversibleRectangle(Rectangle rectangle, Color
15 backColor);
16 [C++] public: static void FillReversibleRectangle(Rectangle rectangle, Color
17 backColor);
18 [VB] Public Shared Sub FillReversibleRectangle(ByVal rectangle As Rectangle,
19 ByVal backColor As Color)
20 [JScript] public static function FillReversibleRectangle(rectangle : Rectangle,
21 backColor : Color);
22

```

### *Description*

Draws a filled, reversible rectangle on the screen.

The *backColor* parameter is used to calculate the fill color of the rectangle so that it is always visible against the background. The **System.Drawing.Rectangle** that represents the dimensions of the rectangle to fill, in screen coordinates. The **System.Drawing.Color** of the background behind the fill.

#### vv) *Light*

```
[C#]      public      static      Color      Light(Color      baseColor);
[C++]      public:      static      Color      Light(Color      baseColor);
[VB] Public Shared Function Light(ByVal baseColor As Color) As Color
[JScript] public static function Light(baseColor : Color) : Color;
```

#### *Description*

Creates a new light color object for the control from the specified color.  
*Return Value:* A **System.Drawing.Color** that represents the light color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to a **System.Drawing.SystemColors.ControlLight** color; otherwise, the color's luminosity value is decreased. The **System.Drawing.Color** to be lightened.

#### ww) *Light*

```
[C#] public static Color Light(Color baseColor, float percOfLightLight);
[C++] public: static Color Light(Color baseColor, float percOfLightLight);
[VB] Public Shared Function Light(ByVal baseColor As Color, ByVal
percOfLightLight As Single) As Color
[JScript] public static function Light(baseColor : Color, percOfLightLight : float) :
Color; Creates a new light color object for the control.
```

## Description

Creates a new light color object for the control from the specified color and lightens it by the specified percentage.

*Return Value:* A **System.Drawing.Color** that represents the light color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to a **System.Drawing.SystemColors.ControlLight** color; otherwise, the color's luminosity value is decreased. The **System.Drawing.Color** to be lightened. The percentage to lighten the specified **System.Drawing.Color**.

## xx) LightLight

[C#]        public        static        Color        LightLight(Color        baseColor);

[C++]       public:        static        Color        LightLight(Color        baseColor);

[VB] Public Shared Function LightLight(ByVal baseColor As Color) As Color

[JScript] public static function LightLight(baseColor : Color) : Color;

## Description

Creates a new light color object for the control from the specified color.

*Return Value:* A **System.Drawing.Color** that represents the light color on the control.

If the specified **System.Drawing.Color** is one of the **System.Drawing.SystemColors**, the color is converted to the **System.Drawing.SystemColors.ControlLightLight** color; otherwise, the color's luminosity value is increased. The **System.Drawing.Color** to be lightened.

ControlStyles enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the style and behavior of the control.

Controls use this enumeration in various properties and methods to specify functionality.

*b) ToString*

|           |         |       |                      |                       |
|-----------|---------|-------|----------------------|-----------------------|
| [C#]      | public  | const | ControlStyles        | AllPaintingInWmPaint; |
| [C++]     | public: | const | ControlStyles        | AllPaintingInWmPaint; |
| [VB]      | Public  | Const | AllPaintingInWmPaint | As ControlStyles      |
| [JScript] | public  | var   | AllPaintingInWmPaint | : ControlStyles;      |

*Description*

The control ignores the window message WM\_ERASEBKGND to reduce flicker. This style should only be used if **System.Windows.Forms.ControlStyles.UserPaint** is set to **true**.

*c) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | ControlStyles | CacheText;       |
| [C++]     | public: | const | ControlStyles | CacheText;       |
| [VB]      | Public  | Const | CacheText     | As ControlStyles |
| [JScript] | public  | var   | CacheText     | : ControlStyles; |

### Description

The control keeps a copy of the text rather than getting it from the **System.Windows.Forms.Control.Handle** each time it is needed. This style defaults to **false**. This behavior improves performance, but makes it difficult to keep the text synchronized.

#### d) ToString

|           |         |       |                  |    |                   |
|-----------|---------|-------|------------------|----|-------------------|
| [C#]      | public  | const | ControlStyles    |    | ContainerControl; |
| [C++]     | public: | const | ControlStyles    |    | ContainerControl; |
| [VB]      | Public  | Const | ContainerControl | As | ControlStyles     |
| [JScript] | public  | var   | ContainerControl | :  | ControlStyles;    |

### Description

The control is a container-like control.

#### e) ToString

|           |         |       |               |    |                |
|-----------|---------|-------|---------------|----|----------------|
| [C#]      | public  | const | ControlStyles |    | DoubleBuffer;  |
| [C++]     | public: | const | ControlStyles |    | DoubleBuffer;  |
| [VB]      | Public  | Const | DoubleBuffer  | As | ControlStyles  |
| [JScript] | public  | var   | DoubleBuffer  | :  | ControlStyles; |

### Description

Drawing is performed in a buffer, and after it completes, the result is output to the screen. Double-buffering prevents flicker caused by the redrawing of the control.

### f) ToString

|           |         |       |                     |                      |
|-----------|---------|-------|---------------------|----------------------|
| [C#]      | public  | const | ControlStyles       | EnableNotifyMessage; |
| [C++]     | public: | const | ControlStyles       | EnableNotifyMessage; |
| [VB]      | Public  | Const | EnableNotifyMessage | As ControlStyles     |
| [JScript] | public  | var   | EnableNotifyMessage | : ControlStyles;     |

#### Description

If **true** , the **System.Windows.Forms.Control.OnNotifyMessage(System.Windows.Forms.Message)** method is called for every message sent to the control's **System.Windows.Forms.Control.WndProc(System.Windows.Forms.Message@)** . This style defaults to **false** .

### g) ToString

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | ControlStyles | FixedHeight;     |
| [C++]     | public: | const | ControlStyles | FixedHeight;     |
| [VB]      | Public  | Const | FixedHeight   | As ControlStyles |
| [JScript] | public  | var   | FixedHeight   | : ControlStyles; |

#### Description

The control has a fixed height.

### h) ToString

|       |         |       |               |                  |
|-------|---------|-------|---------------|------------------|
| [C#]  | public  | const | ControlStyles | FixedWidth;      |
| [C++] | public: | const | ControlStyles | FixedWidth;      |
| [VB]  | Public  | Const | FixedWidth    | As ControlStyles |

```
[JScript]      public      var      FixedWidth      :      ControlStyles;
```

### *Description*

The control has a fixed width.

#### *i) ToString*

```
[C#]           public      const      ControlStyles      Opaque;
```

```
[C++]          public:      const      ControlStyles      Opaque;
```

```
[VB]           Public      Const      Opaque      As      ControlStyles
```

```
[JScript]      public      var      Opaque      :      ControlStyles;
```

### *Description*

The control is drawn opaque and the background is not painted.

#### *j) ToString*

```
[C#]           public      const      ControlStyles      ResizeRedraw;
```

```
[C++]          public:      const      ControlStyles      ResizeRedraw;
```

```
[VB]           Public      Const      ResizeRedraw      As      ControlStyles
```

```
[JScript]      public      var      ResizeRedraw      :      ControlStyles;
```

### *Description*

The control is redrawn when it is resized.

#### *k) ToString*

```
[C#]           public      const      ControlStyles      Selectable;
```



|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C++]     | public: | const | ControlStyles | Selectable;      |
| [VB]      | Public  | Const | Selectable    | As ControlStyles |
| [JScript] | public  | var   | Selectable    | : ControlStyles; |

*Description*

The control can receive focus.

*l) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | ControlStyles | StandardClick;   |
| [C++]     | public: | const | ControlStyles | StandardClick;   |
| [VB]      | Public  | Const | StandardClick | As ControlStyles |
| [JScript] | public  | var   | StandardClick | : ControlStyles; |

*Description*

The control implements the standard **System.Windows.Forms.Control.Click** behavior.

*m) ToString*

|           |         |       |                     |                      |
|-----------|---------|-------|---------------------|----------------------|
| [C#]      | public  | const | ControlStyles       | StandardDoubleClick; |
| [C++]     | public: | const | ControlStyles       | StandardDoubleClick; |
| [VB]      | Public  | Const | StandardDoubleClick | As ControlStyles     |
| [JScript] | public  | var   | StandardDoubleClick | : ControlStyles;     |

*Description*

The control implements the standard

**System.Windows.Forms.Control.DoubleClick** behavior. This style is ignored if **System.Windows.Forms.ControlStyles.StandardClick** is not set.

*n) ToString*

[C#] public const ControlStyles SupportsTransparentBackColor;

[C++] public: const ControlStyles SupportsTransparentBackColor;

[VB] Public Const SupportsTransparentBackColor As ControlStyles

[JScript] public var SupportsTransparentBackColor : ControlStyles;

*Description*

The control accepts a **System.Windows.Forms.Control.BackColor** with an alpha component of less than 255 to simulate transparency. Transparency will be simulated only if the **System.Windows.Forms.ControlStyles.UserPaint** bit is set to **true** and the parent control is derived from

**System.Windows.Forms.Control** .

*o) ToString*

[C#] public const ControlStyles UserMouse;

[C++] public: const ControlStyles UserMouse;

[VB] Public Const UserMouse As ControlStyles

[JScript] public var UserMouse : ControlStyles;

*Description*

The control does its own mouse processing, and mouse events are not handled by the operating system.

*p) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | ControlStyles | UserPaint;       |
| [C++]     | public: | const | ControlStyles | UserPaint;       |
| [VB]      | Public  | Const | UserPaint     | As ControlStyles |
| [JScript] | public  | var   | UserPaint     | : ControlStyles; |

*Description*

The control paints itself rather than the operating system doing so. This style only applies to classes derived from **System.Windows.Forms.Control**.

ConvertEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **System.Windows.Forms.Binding.Format** and **System.Windows.Forms.Binding.Parse** events.

The **System.Windows.Forms.ConvertEventArgs** is used to format and unformat values displayed by a Windows Forms control that is databound through a **System.Windows.Forms.Binding** object. The **System.Windows.Forms.Binding.Format** event occurs whenever a control property is bound to a value, and the **System.Windows.Forms.Binding.Parse** event occurs whenever the bound value changes.

*b) ConvertEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#]    public    ConvertEventArgs(object    value,    Type    desiredType);
[C++]    public:    ConvertEventArgs(Object*    value,    Type*    desiredType);
[VB]    Public Sub New(ByVal value As Object, ByVal desiredType As Type)
[JavaScript]    public function ConvertEventArgs(value : Object, desiredType : Type);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ConvertEventArgs** class. An **System.Object** that contains the value of the current property. The **System.Type** of the value.

d) *DesiredType*

e) *ToString*

```
[C#]            public            Type            DesiredType            {get;}
[C++]            public:            __property            Type*            get_DesiredType();
[VB]            Public            ReadOnly            Property            DesiredType            As            Type
[JavaScript]            public            function            get            DesiredType()            :            Type;
```

*Description*

Gets the data type of the desired value.

The **System.Windows.Forms.ConvertEventArgs.DesiredType** property enables you to check the type of the property that the value is being converted to.

f) *Value*

g) *ToString*

[C#] public object Value {get; set;}

[C++] public: \_\_property Object\* get\_Value();public: \_\_property void  
set\_Value(Object\*);

[VB] Public Property Value As Object

[JScript] public function get Value() : Object;public function set Value(Object);

### *Description*

Gets or sets the value of the **System.Windows.Forms.ConvertEventArgs** object.

The value contained by the **System.Windows.Forms.ConvertEventArgs.Value** property depends on the event in which the **System.Windows.Forms.ConvertEventArgs** object is returned. The **System.Windows.Forms.ConvertEventArgs** object can be returned in either the the **System.Windows.Forms.Binding.Format** event, or the **System.Windows.Forms.Binding.Parse** event.

ConvertEventHandler delegate (System.Windows.Forms)

a) *ToString*

### *Description*

Represents the method that will handle the **System.Windows.Forms.Binding.Parse** and **System.Windows.Forms.Binding.Format** events of a **System.Windows.Forms.Binding** object. The source of the event. A **System.Windows.Forms.ConvertEventArgs** that contains the event data.

When you create a(n) **System.Windows.Forms.ConvertEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The

event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

CreateParams class (System.Windows.Forms)

*a) ToString*

### Description

Encapsulates the information needed when creating a control.

The information in a **System.Windows.Forms.CreateParams** object can be used to pass information about the initial state and appearance of a control. Most **System.Windows.Forms.Control** derived controls override the **System.Windows.Forms.Control.CreateParams** property to pass in the appropriate values or include additional information in the **System.Windows.Forms.CreateParams** object.

*b) CreateParams*

*Example Syntax:*

*c) ToString*

[C#] public CreateParams();

[C++] public: CreateParams();

[VB] Public Sub New()

[JScript] public function CreateParams();

*d) Caption*

*e) ToString*

[C#] public string Caption {get; set;}

[C++] public: \_\_property String\* get\_Caption();public: \_\_property void

```

1 set_Caption(String*);
2 [VB]      Public      Property      Caption      As      String
3 [JScript] public function get Caption() : String;public function set Caption(String);
4

```

#### *Description*

Gets or sets the control's initial text.

*f)      ClassName*

*g)      ToString*

```

10 [C#]      public      string      ClassName      {get;      set;}
11 [C++] public: __property String* get_ClassName();public: __property void
12 set_ClassName(String*);
13 [VB]      Public      Property      ClassName      As      String
14 [JScript] public function get ClassName() : String;public function set
15 ClassName(String);
16

```

#### *Description*

Gets or sets the name of the class to derive the control from.

The default value for this property is **null** , indicating that the control is not derived from an existing control class. To derive from an existing control class, store the system class name in this property. For example, to derive from the standard **System.Windows.Forms.Button** control, set this property to "BUTTON".

h) *ClassStyle*

i) *ToString*

[C#] public int ClassStyle {get; set;}

[C++] public: \_\_property int get\_ClassStyle();public: \_\_property void set\_ClassStyle(int);

[VB] Public Property ClassStyle As Integer

[JScript] public function get ClassStyle() : int;public function set ClassStyle(int);

#### *Description*

Gets or sets a bitwise combination of class style values.

The **System.Windows.Forms.CreateParams.ClassStyle** property is ignored if the **System.Windows.Forms.CreateParams.ClassName** property is not **null**.

j) *ExStyle*

k) *ToString*

[C#] public int ExStyle {get; set;}

[C++] public: \_\_property int get\_ExStyle();public: \_\_property void set\_ExStyle(int);

[VB] Public Property ExStyle As Integer

[JScript] public function get ExStyle() : int;public function set ExStyle(int);

#### *Description*

Gets or sets a bitwise combination of extended window style values.



The **System.Windows.Forms.CreateParams.ExStyle** property supports extended appearance and initial state values to apply to the control.

*l) Height*

*m) ToString*

```
[C#]          public          int          Height          {get;          set;}
```

```
[C++] public: __property int get_Height();public: __property void set_Height(int);
```

```
[VB]          Public          Property          Height          As          Integer
```

```
[JScript] public function get Height() : int;public function set Height(int);
```

#### *Description*

Gets or sets the initial height of the control.

*n) Param*

*o) ToString*

```
[C#]          public          object          Param          {get;          set;}
```

```
[C++] public: __property Object* get_Param();public: __property void  
set_Param(Object*);
```

```
[VB]          Public          Property          Param          As          Object
```

```
[JScript] public function get Param() : Object;public function set Param(Object);
```

#### *Description*

Gets or sets additional parameter information needed to create the control.

If you are creating a multiple document interface (MDI) client window, the **System.Windows.Forms.CreateParams.Param** property must reference a **CLIENTCREATESTRUCT** structure.

*p) Parent*

*q) ToString*

```
[C#]      public      IntPtr      Parent      {get;      set;}
[C++]    public:  __property  IntPtr  get_Parent();public:  __property  void
set_Parent(IntPtr);
[VB]      Public      Property      Parent      As      IntPtr
[JScript] public function get Parent() : IntPtr;public function set Parent(IntPtr);
```

*Description*

Gets or sets the control's parent.

*r) Style*

*s) ToString*

```
[C#]      public      int      Style      {get;      set;}
[C++]    public:  __property  int  get_Style();public:  __property  void  set_Style(int);
[VB]      Public      Property      Style      As      Integer
[JScript] public function get Style() : int;public function set Style(int);
```

*Description*

Gets or sets a bitwise combination of window style values.

The **System.Windows.Forms.CreateParams.Style** property controls the appearance of the control and its initial state.

t) *Width*

u) *ToString*

[C#] public int Width {get; set;}

[C++] public: \_\_property int get\_Width();public: \_\_property void set\_Width(int);

[VB] Public Property Width As Integer

[JScript] public function get Width() : int;public function set Width(int);

*Description*

Gets or sets the initial width of the control.

v) *X*

w) *ToString*

[C#] public int X {get; set;}

[C++] public: \_\_property int get\_X();public: \_\_property void set\_X(int);

[VB] Public Property X As Integer

[JScript] public function get X() : int;public function set X(int);

*Description*

Gets or sets the initial left position of the control.

x) *Y*

y) *ToString*

[C#] public int Y {get; set;}

[C++] public: \_\_property int get\_Y();public: \_\_property void set\_Y(int);

[VB]            Public            Property            Y            As            Integer

[JScript]   public   function   get   Y()   :   int;public   function   set   Y(int);

*Description*

Gets or sets the top position of the initial location of the control.

*z) ToString*

[C#]            public            override            string            ToString();

[C++]            public:            String\*            ToString();

[VB]    Overrides    Public    Function    ToString()    As    String

[JScript]   public   override   function   ToString()   :   String;

*Description*

CurrencyManager class (System.Windows.Forms)

*a) ToString*

*Description*

Manages a list of **System.Windows.Forms.Binding** objects.

The **System.Windows.Forms.CurrencyManager** derives from the **System.Windows.Forms.BindingManagerBase** class. Use the **System.Windows.Forms.BindingContext** to return either a **System.Windows.Forms.CurrencyManager** or a **System.Windows.Forms.PropertyManager** . The actual object returned depends on the data source and data member passed to the **System.Windows.Forms.BindingContext.Item(System.Object)** property of the **System.Windows.Forms.BindingContext** . If the data source is an

object that can only return a single property (instead of a list of objects), the type will be a **System.Windows.Forms.PropertyManager** . For example, if you specify a **System.Windows.Forms.TextBox** as the data source, a **System.Windows.Forms.PropertyManager** will be returned. If, on the other hand, the data source is an object that implements **System.Collections.IList** , **System.ComponentModel.IListSource** , or **System.ComponentModel.IBindingList** , a **System.Windows.Forms.CurrencyManager** will be returned.

*b) ToString*

|           |            |               |            |
|-----------|------------|---------------|------------|
| [C#]      | protected  | Type          | finalType; |
| [C++]     | protected: | Type*         | finalType; |
| [VB]      | Protected  | finalType     | As Type    |
| [JScript] | protected  | var finalType | : Type;    |

*Description*

Specifies the data type of the list.

*c) ToString*

|           |            |                  |               |
|-----------|------------|------------------|---------------|
| [C#]      | protected  | int              | listposition; |
| [C++]     | protected: | int              | listposition; |
| [VB]      | Protected  | listposition     | As Integer    |
| [JScript] | protected  | var listposition | : int;        |

*Description*

Specifies that there are no list members.

d) *Bindings*

e) *Count*

f) *ToString*

#### *Description*

Gets the number of items in the list.

Use the count property to determine when the end of a list has been reached.

Because the **System.Windows.Forms.CurrencyManager** maintains a 0-based array of items, the end of the list is always

**System.Windows.Forms.CurrencyManager.Count** minus one.

g) *Current*

h) *ToString*

[C#]        public        override        object        Current        {get;}

[C++]       public:       \_\_property       virtual       Object\*       get\_Current();

[VB]       Overrides       Public       ReadOnly       Property       Current       As       Object

[JScript]       public       function       get       Current()       :       Object;

#### *Description*

Gets the current item in the list.

In order to get the current item, you must know its data type in order to cast it correctly. For example, if the data source is a **System.Data.DataView** or

**System.Data.DataTable**, you must cast the current item as a **System.Data.DataRowView** object.

i) *List*

j) *ToString*

```
[C#]          public          IList          List          {get;}
[C++]          public:          __property          IList*          get_List();
[VB]          Public          ReadOnly          Property          List          As          IList
[JScript]          public          function          get          List()          :          IList;
```

#### *Description*

Gets the IList for this **System.Windows.Forms.CurrencyManager** .

If the returned IList is of type Gets the list as an object.

k) *Position*

l) *ToString*

```
[C#]          public          override          int          Position          {get;          set;}
[C++]          public:          __property          virtual int          get_Position();public:          __property          virtual void
set_Position(int);
[VB]          Overrides          Public          Property          Position          As          Integer
[JScript]          public          function          get          Position()          :          int;public          function          set          Position(int);
```

#### *Description*

Gets or sets the position you are at within the list.

An important property of the **System.Windows.Forms.CurrencyManager** is the **System.Windows.Forms.CurrencyManager.Position** property. In a list of items, you can only view one item out of the entire list. To determine which item is viewed, set the **System.Windows.Forms.CurrencyManager.Position** to a number between 0 (the beginning of the list) and

**System.Windows.Forms.CurrencyManager.Count** minus one (the end of the list).

*m) ToString*

*Description*

Occurs when the current item has been altered.

The **System.Windows.Forms.CurrencyManager.ItemChanged** method event is supported only if the underlying data source implements the **System.ComponentModel.IBindingList** interface.

*n) AddNew*

|           |           |          |          |           |
|-----------|-----------|----------|----------|-----------|
| [C#]      | public    | override | void     | AddNew(); |
| [C++]     | public:   |          | void     | AddNew(); |
| [VB]      | Overrides | Public   | Sub      | AddNew()  |
| [JScript] | public    | override | function | AddNew(); |

*Description*

Adds a new item to the underlying list.

This method is only supported if the data source implements **System.ComponentModel.IBindingList** and the **System.ComponentModel.IBindingList.AllowNew** property returns **true**.

*o) CancelCurrentEdit*

|       |           |          |      |                      |
|-------|-----------|----------|------|----------------------|
| [C#]  | public    | override | void | CancelCurrentEdit(); |
| [C++] | public:   |          | void | CancelCurrentEdit(); |
| [VB]  | Overrides | Public   | Sub  | CancelCurrentEdit()  |



|           |        |          |          |                      |
|-----------|--------|----------|----------|----------------------|
| [JScript] | public | override | function | CancelCurrentEdit(); |
|-----------|--------|----------|----------|----------------------|

### Description

Cancels the current edit operation.

This method is supported only if the objects contained by the data source implement the **System.ComponentModel.IEditableObject** interface. If the objects contained within the data source do not implement the **System.ComponentModel.IEditableObject** interface, changes made to the data will not be discarded.

### p) CheckEmpty

|           |            |          |               |
|-----------|------------|----------|---------------|
| [C#]      | protected  | void     | CheckEmpty(); |
| [C++]     | protected: | void     | CheckEmpty(); |
| [VB]      | Protected  | Sub      | CheckEmpty()  |
| [JScript] | protected  | function | CheckEmpty(); |

### Description

Throws an exception if there is no list.

### q) EndCurrentEdit

|           |           |          |                   |                   |
|-----------|-----------|----------|-------------------|-------------------|
| [C#]      | public    | override | void              | EndCurrentEdit(); |
| [C++]     | public:   | void     | EndCurrentEdit(); |                   |
| [VB]      | Overrides | Public   | Sub               | EndCurrentEdit()  |
| [JScript] | public    | override | function          | EndCurrentEdit(); |

### Description

Ends the current edit operation.

This method is supported only if the objects contained by the data source implement the **System.ComponentModel.IEditableObject** interface.

*r) GetItemProperties*

```
[C#] public override PropertyDescriptorCollection GetItemProperties();
[C++] public: PropertyDescriptorCollection* GetItemProperties();
[VB] Overrides Public Function GetItemProperties() As
PropertyDescriptorCollection
[JScript] public override function GetItemProperties() :
PropertyDescriptorCollection; Gets the
System.ComponentModel.PropertyDescriptorCollection for the list.
```

*Description*

Gets the **System.ComponentModel.PropertyDescriptorCollection** for the list.

*Return Value:* A **System.ComponentModel.PropertyDescriptorCollection** for the list.

The **System.ComponentModel.PropertyDescriptorCollection** for the **System.Windows.Forms.CurrencyManager** is used to specify a column in the list.

*s) GetListName*

```
[C#] protected internal override string GetListName(ArrayList listAccessors);
[C++] protected public: String* GetListName(ArrayList* listAccessors);
[VB] Overrides Protected Friend Dim Function GetListName(ByVal listAccessors
As ArrayList) As String
[JScript] package override function GetListName(listAccessors : ArrayList) :
String;
```

## *Description*

Gets the name of the specified list.

*Return Value:* The name of the list. The list, typed as **System.Collections.ArrayList**, that you are interested in.

### *t) OnCurrentChanged*

[C#] protected internal override void OnCurrentChanged(EventArgs e);

[C++] protected public: void OnCurrentChanged(EventArgs\* e);

[VB] Overrides Protected Friend Dim Sub OnCurrentChanged(ByVal e As EventArgs)

[JScript] package override function OnCurrentChanged(e : EventArgs);

## *Description*

Raises the

**System.Windows.Forms.BindingManagerBase.CurrentChanged** event.

An **System.EventArgs** that contains the event data.

### *u) OnItemChanged*

[C#] protected virtual void OnItemChanged(ItemChangedEventArgs e);

[C++] protected: virtual void OnItemChanged(ItemChangedEventArgs\* e);

[VB] Overridable Protected Sub OnItemChanged(ByVal e As ItemChangedEventArgs)

[JScript] protected function OnItemChanged(e : ItemChangedEventArgs);

## *Description*

Raises the **System.Windows.Forms.CurrencyManager.ItemChanged** event. An **System.Windows.Forms.ItemChangedEventArgs** that contains the event data.

*v) OnPositionChanged*

[C#] protected virtual void OnPositionChanged(EventArgs e);

[C++] protected: virtual void OnPositionChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnPositionChanged(ByVal e As EventArgs)

[JScript] protected function OnPositionChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.BindingManagerBase.PositionChanged** event. An **System.EventArgs** that contains the event data.

*w) Refresh*

[C#] public void Refresh();

[C++] public: void Refresh();

[VB] Public Sub Refresh()

[JScript] public function Refresh();

*Description*

Forces a repopulation of the bound controls.

Use the **System.Windows.Forms.CurrencyManager.Refresh** method when the data source doesn't support notification when it is changed, for example an **System.Array**.

x) *RemoveAt*

```
[C#]      public      override      void      RemoveAt(int      index);
[C++]      public:      void      RemoveAt(int      index);
[VB]      Overrides      Public      Sub      RemoveAt(ByVal      index      As      Integer)
[JScript]      public      override      function      RemoveAt(index      :      int);
```

*Description*

Removes the item at the specified index.

This method was designed to allow complex controls, such as the **System.Windows.Forms.DataGrid** control, to remove items from the list. It is not recommended that you use this method to actually remove items. Instead, use the **System.Data.DataView.Delete** method of the **System.Data.DataView** class to delete items. The index of the item to remove from the list.

y) *ResumeBinding*

```
[C#]      public      override      void      ResumeBinding();
[C++]      public:      void      ResumeBinding();
[VB]      Overrides      Public      Sub      ResumeBinding()
[JScript]      public      override      function      ResumeBinding();
```

*Description*

Resumes binding of component properties to list items.

z) *SuspendBinding*

```
[C#]      public      override      void      SuspendBinding();
```

|           |           |          |                   |                   |
|-----------|-----------|----------|-------------------|-------------------|
| [C++]     | public:   | void     | SuspendBinding(); |                   |
| [VB]      | Overrides | Public   | Sub               | SuspendBinding()  |
| [JScript] | public    | override | function          | SuspendBinding(); |

#### Description

Suspends binding.

#### aa) UpdateIsBinding

|           |            |           |                    |                    |
|-----------|------------|-----------|--------------------|--------------------|
| [C#]      | protected  | override  | void               | UpdateIsBinding(); |
| [C++]     | protected: | void      | UpdateIsBinding(); |                    |
| [VB]      | Overrides  | Protected | Sub                | UpdateIsBinding()  |
| [JScript] | protected  | override  | function           | UpdateIsBinding(); |

#### Description

Updates the status of the binding.

Cursor class (System.Windows.Forms)

#### a) UpdateIsBinding

#### Description

Represents the image used to paint the mouse pointer.

A cursor is a small picture whose location on the screen is controlled by a pointing device, such as a mouse, pen, or trackball. When the user moves the pointing device, the operating system moves the cursor accordingly.

**b) Cursor**

*Example Syntax:*

**c) UpdateIsBinding**

```
[C#]           public           Cursor(IntPtr           handle);
[C++]           public:           Cursor(IntPtr           handle);
[VB]   Public   Sub   New(ByVal   handle   As   IntPtr)
[JScript] public function Cursor(handle : IntPtr); Initializes a new instance of the
System.Windows.Forms.Cursor class.
```

*Description*

Initializes a new instance of the **System.Windows.Forms.Cursor** class from the specified Windows handle.

You must free the cursor handle when you are done with it. For more information about disposing of resources, see . The Windows handle of the cursor to create.

**d) Cursor**

*Example Syntax:*

**e) UpdateIsBinding**

```
[C#]           public           Cursor(Stream           stream);
[C++]           public:           Cursor(Stream*           stream);
[VB]   Public   Sub   New(ByVal   stream   As   Stream)
[JScript]   public   function   Cursor(stream   :   Stream);
```

*Description*

1 Initializes a new instance of the **System.Windows.Forms.Cursor** class from  
the specified data stream.

2 The data stream specified by *stream* must contain a cursor (.cur) file. The data  
stream to load the **System.Windows.Forms.Cursor** from.

4 *f) Cursor*

5 *Example Syntax:*

6 *g) UpdateIsBinding*

8 [C#] public Cursor(string fileName);

9 [C++] public: Cursor(String\* fileName);

10 [VB] Public Sub New(ByVal fileName As String)

11 [JScript] public function Cursor(fileName : String);

13 *Description*

14 Initializes a new instance of the **System.Windows.Forms.Cursor** class from  
the specified file.

15 The *fileName* parameter must reference a standard cursor (.cur) file. The cursor  
file to load.

17 *h) Cursor*

18 *Example Syntax:*

19 *i) UpdateIsBinding*

21 [C#] public Cursor(Type type, string resource);

22 [C++] public: Cursor(Type\* type, String\* resource);

23 [VB] Public Sub New(ByVal type As Type, ByVal resource As String)

24 [JScript] public function Cursor(type : Type, resource : String);



## Description

Initializes a new instance of the **System.Windows.Forms.Cursor** class from the specified resource with the specified resource type. The resource **System.Type** . A **System.String** that represents the name of the resource.

j) *Clip*

k) *UpdateIsBinding*

```
[C#] public static Rectangle Clip {get; set;}
```

```
[C++] public: __property static Rectangle get_Clipo();public: __property static void  
set_Clipo(Rectangle);
```

```
[VB] Public Shared Property Clip As Rectangle
```

```
[JScript] public static function get Clip() : Rectangle;public static function set  
Clip(Rectangle);
```

## Description

Gets or sets the bounds that represents the clipping rectangle for the cursor.

A clipped cursor is allowed to move only within its clipping rectangle. Generally, the system allows this only if the mouse is currently captured. If the cursor is not currently clipped, the resulting rectangle contains the dimensions of the entire screen.

l) *Current*

m) *UpdateIsBinding*

```
[C#] public static Cursor Current {get; set;}
```

```
[C++] public: __property static Cursor* get_Current();public: __property static  
void set_Current(Cursor*);
```

```

1 [VB]      Public      Shared      Property      Current      As      Cursor
2 [JScript] public static function get Current() : Cursor;public static function set
3 Current(Cursor);
4

```

### Description

Gets or sets a cursor object that represents the mouse cursor.

Setting the **System.Windows.Forms.Cursor.Current** property changes the cursor currently displayed, and the application stops listening for mouse events. For example, you might set the **System.Windows.Forms.Cursor.Current** property to **System.Windows.Forms.Cursors.WaitCursor** before you start filling a **System.Windows.Forms.TreeView** , **System.Windows.Forms.DataGrid** , or **System.Windows.Forms.ListBox** control with a large amount of data. After the loop is completed, set this property back to **System.Windows.Forms.Cursors.Default** to display the appropriate cursor for each control.

*n) Handle*

*o) UpdateIsBinding*

```

15 [C#]      public      IntPtr      Handle      {get;}
16 [C++]      public:      __property      IntPtr      get_Handle();
17 [VB]      Public      ReadOnly      Property      Handle      As      IntPtr
18 [JScript]      public      function      get      Handle()      :      IntPtr;
19

```

### Description

Gets the handle of the cursor.

This is not a copy of the handle; do not dispose of it.

**p) Position**

**q) UpdateIsBinding**

[C#] public static Point Position {get; set;}

[C++] public: \_\_property static Point get\_Position();public: \_\_property static void set\_Position(Point);

[VB] Public Shared Property Position As Point

[JScript] public static function get Position() : Point;public static function set Position(Point);

**Description**

Gets or sets the cursor's position.

**r) Size**

**s) UpdateIsBinding**

[C#] public Size Size {get;}

[C++] public: \_\_property Size get\_Size();

[VB] Public ReadOnly Property Size As Size

[JScript] public function get Size() : Size;

**Description**

Gets the size of the cursor object.

**t) CopyHandle**

[C#] public IntPtr CopyHandle();

```

1  [C++]          public:          IntPtr          CopyHandle();
2  [VB]           Public          Function          CopyHandle()      As          IntPtr
3  [JScript]      public          function          CopyHandle()      :          IntPtr;

```

#### *Description*

Copies the handle of this **System.Windows.Forms.Cursor** .

*Return Value:* An **System.IntPtr** object that represents the cursor's handle.

The handle created as a result of calling this method must be disposed of when you are done with it because it will not be disposed of by the garbage collector.

#### *u) Dispose*

```

11 [C#]           public          void          Dispose();
12 [C++]          public:          __sealed          void          Dispose();
13 [VB]           NotOverridable          Public          Sub          Dispose()
14 [JScript]      public          function          Dispose();

```

#### *Description*

Releases all resources used by the **System.Windows.Forms.Cursor** .

Calling **System.Windows.Forms.Cursor.Dispose** allows the resources used by the **System.Windows.Forms.Cursor** to be reallocated for other purposes. For more information about **System.Windows.Forms.Cursor.Dispose** , see .

#### *v) Draw*

```

22 [C#]   public   void   Draw(Graphics   g,   Rectangle   targetRect);
23 [C++]   public:   void   Draw(Graphics*   g,   Rectangle   targetRect);
24 [VB]   Public Sub Draw(ByVal g As Graphics, ByVal targetRect As Rectangle)

```

[JScript] public function Draw(g : Graphics, targetRect : Rectangle);

### *Description*

Draws the cursor on the specified surface, within the specified bounds.

The drawing command originates on the *g***System.Drawing.Graphics** object, but a **System.Drawing.Graphics** object does not contain information about how to render a given image, so it passes the call to the **System.Windows.Forms.Cursor** object. The **System.Windows.Forms.Cursor.Draw(System.Drawing.Graphics, System.Drawing.Rectangle)** method crops the image to the given dimensions and allows you to specify a **System.Drawing.Rectangle** within which to draw the **System.Windows.Forms.Cursor**. The **System.Drawing.Graphics** surface on which to draw the **System.Windows.Forms.Cursor**. The **System.Drawing.Rectangle** that represents the bounds of the **System.Windows.Forms.Cursor**.

### *w) DrawStretched*

[C#] public void DrawStretched(Graphics g, Rectangle targetRect);

[C++] public: void DrawStretched(Graphics\* g, Rectangle targetRect);

[VB] Public Sub DrawStretched(ByVal g As Graphics, ByVal targetRect As Rectangle)

[JScript] public function DrawStretched(g : Graphics, targetRect : Rectangle);

### *Description*

Draws the cursor in a stretched format on the specified surface, within the specified bounds.

The drawing command originates on the *g***System.Drawing.Graphics** object, but a **System.Drawing.Graphics** object does not contain information about how to render a given image, so it passes the call to the **System.Windows.Forms.Cursor** object. The **System.Windows.Forms.Cursor.DrawStretched(System.Drawing.Graphics, System.Drawing.Rectangle)** method stretches the image to fill the specified **System.Drawing.Rectangle** when the cursor is drawn. The

**System.Drawing.Graphics** surface on which to draw the **System.Windows.Forms.Cursor** . The **System.Drawing.Rectangle** that represents the bounds of the **System.Windows.Forms.Cursor** .

*x) Equals*

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object\* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

*Description*

Returns a value indicating whether the cursor is equal to the specified **System.Windows.Forms.Cursor** object.

*Return Value:* **true** if the cursor is equal to the specified **System.Windows.Forms.Cursor** object; otherwise, **false** . The **System.Windows.Forms.Cursor** object to compare.

*y) Finalize*

[C#] ~Cursor();

[C++] ~Cursor();

[VB] Overrides Protected Sub Finalize()

[JScript] protected override function Finalize(); Cleans up Windows resources for this object.

*z) GetHashCode*

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

#### Description

##### aa) Hide

[C#] public static void Hide();

[C++] public: static void Hide();

[VB] Public Shared Sub Hide()

[JScript] public static function Hide();

#### Description

Hides the cursor.

The **System.Windows.Forms.Cursor.Show** and **System.Windows.Forms.Cursor.Hide** method calls must be balanced. For every call to the **System.Windows.Forms.Cursor.Hide** method there must be a corresponding call to the **System.Windows.Forms.Cursor.Show** method.

##### bb) op\_Equality

[C#] public static bool operator ==(Cursor left, Cursor right);

[C++] public: static bool op\_Equality(Cursor\* left, Cursor\* right);

[VB] returnValue = Cursor.op\_Equality(left, right)

[JScript] returnValue = left == right;

#### Description

Returns a value indicating whether two instances of a

**System.Windows.Forms.Cursor** object are equal.

*Return Value:* **true** if two instances of a **System.Windows.Forms.Cursor** object are equal; otherwise, **false** . A **System.Windows.Forms.Cursor** object to compare. A **System.Windows.Forms.Cursor** object to compare.

*cc) op\_Inequality*

[C#] public static bool operator !=(Cursor left, Cursor right);

[C++] public: static bool op\_Inequality(Cursor\* left, Cursor\* right);

[VB] returnValue = Cursor.op\_Inequality(left, right)

[JScript] returnValue = left != right;

*Description*

Returns a value indicating whether two instances of a

**System.Windows.Forms.Cursor** object are not equal.

*Return Value:* **true** if two instances of a **System.Windows.Forms.Cursor** object are not equal; otherwise, **false** . A **System.Windows.Forms.Cursor** object to compare. A **System.Windows.Forms.Cursor** object to compare.

*dd) Show*

[C#] public static void Show();

[C++] public: static void Show();

[VB] Public Shared Sub Show()

[JScript] public static function Show();

*Description*

Displays the cursor.

The **System.Windows.Forms.Cursor.Show** and

**System.Windows.Forms.Cursor.Hide** method calls must be balanced. For



every call to the **System.Windows.Forms.Cursor.Hide** method there must be a corresponding call to the **System.Windows.Forms.Cursor.Show** method.

*ee) ISerializable.GetObjectData*

[C#] void ISerializable.GetObjectData(SerializationInfo si, StreamingContext context);

[C++] void ISerializable::GetObjectData(SerializationInfo\* si, StreamingContext context);

[VB] Sub GetObjectData(ByVal si As SerializationInfo, ByVal context As StreamingContext) Implements ISerializable.GetObjectData

[JScript] function ISerializable.GetObjectData(si : SerializationInfo, context : StreamingContext);

*ff) ToString*

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

*Description*

Retrieves a human readable string representing this **System.Windows.Forms.Cursor**.

*Return Value:* A **System.String** that represents this **System.Windows.Forms.Cursor**.

CursorConverter class (System.Windows.Forms)

*a) ToString*

*Description*

Provides a type converter to convert **System.Windows.Forms.Cursor** objects to and from various other representations.

For more information about type converters, see the **System.ComponentModel.TypeConverter** base class and .

*b) CursorConverter*

*Example Syntax:*

*c) ToString*

[C#] public CursorConverter();

[C++] public: CursorConverter();

[VB] Public Sub New()

[JScript] public function CursorConverter();

*d) CanConvertFrom*

[C#] public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType);

[C++] public: bool CanConvertFrom(ITypeDescriptorContext\* context, Type\* sourceType);

[VB] Overrides Public Function CanConvertFrom(ByVal context As ITypeDescriptorContext, ByVal sourceType As Type) As Boolean

```

1 [JScript]    public    override    function    CanConvertFrom(context    :
2 ITypeDescriptorContext,    sourceType    :    Type)    :    Boolean;

```

#### 4 *Description*

5 Determines if this converter can convert an object in the given source type to the native type of the converter.

6 *Return Value:* True if this object can perform the conversion. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null. The type you wish to convert from.

#### 9 e) *CanConvertTo*

```

11 [C#] public override bool CanConvertTo(ITypeDescriptorContext context, Type
12 destinationType);

```

```

13 [C++] public: bool CanConvertTo(ITypeDescriptorContext* context, Type*
14 destinationType);

```

```

15 [VB] Overrides Public Function CanConvertTo(ByVal context As
16 ITypeDescriptorContext, ByVal destinationType As Type) As Boolean

```

```

17 [JScript]    public    override    function    CanConvertTo(context    :
18 ITypeDescriptorContext,    destinationType    :    Type)    :    Boolean;

```

#### 20 *Description*

21 Gets a value indicating whether this converter can convert an object to the given destination type using the context.

22 *Return Value:* **true** if this converter can perform the conversion; otherwise, **false** .

24 The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null** , so always check. Also, properties on the context object can return **null** . An

**System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you wish to convert to.

*f) ConvertFrom*

```
[C#] public override object ConvertFrom(ITypeDescriptorContext context,
CultureInfo culture, object value);
```

```
[C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
CultureInfo* culture, Object* value);
```

```
[VB] Overrides Public Function ConvertFrom(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
As Object
```

```
[JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object) : Object;
```

*Description*

Converts the given object to the converter's native type.

*g) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
```

```
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
culture, Object* value, Type* destinationType);
```

```
[VB] Overrides Public Function ConvertTo(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
ByVal destinationType As Type) As Object
```

```

1 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,
2 culture : CultureInfo, value : Object, destinationType : Type) : Object;
3

```

#### 4 *Description*

5 Converts the given object to another type. The most common types to convert  
6 are to and from a string object. The default implementation will make a call to  
7 ToString on the object if the object is valid and if the destination type is string. If  
8 this cannot convert to the destination type, this will throw a  
9 NotSupportedException.

#### 8 *h) GetStandardValues*

```

10 [C#]          public          override          StandardValuesCollection
11 GetStandardValues(ITypeDescriptorContext          context);
12 [C++]          public:          StandardValuesCollection*
13 GetStandardValues(ITypeDescriptorContext*          context);
14 [VB] Overrides Public Function GetStandardValues(ByVal context As
15 ITypeDescriptorContext)          As          StandardValuesCollection
16 [JScript] public override function GetStandardValues(context :
17 ITypeDescriptorContext)          :          StandardValuesCollection;
18

```

#### 19 *Description*

20 Retrieves a collection containing a set of standard values for the data type this  
21 validator is designed for. This will return null if the data type does not support a  
22 standard set of values.

22 *Return Value:* A collection containing a standard set of valid values, or null. The  
23 default implementation always returns null. A formatter context. This object can  
24 be used to extract additional information about the environment this converter is  
25 being invoked from. This may be null, so you should always check. Also,  
26 properties on the context object may also return null.

i) *GetStandardValuesSupported*

[C#] public override bool GetStandardValuesSupported(ITypeDescriptorContext context);

[C++] public: bool GetStandardValuesSupported(ITypeDescriptorContext\* context);

[VB] Overrides Public Function GetStandardValuesSupported(ByVal context As ITypeDescriptorContext) As Boolean

[JScript] public override function GetStandardValuesSupported(context : ITypeDescriptorContext) : Boolean;

*Description*

Determines if this object supports a standard set of values that can be picked from a list.

*Return Value:* Returns true if GetStandardValues should be called to find a common set of values the object supports. A type descriptor through which additional context may be provided.

Cursors class (System.Windows.Forms)

a) *ToString*

*Description*

Provides a collection of **System.Windows.Forms.Cursor** objects for use by a Windows Forms application.

Some of the **System.Windows.Forms.Cursor** objects in this class can take on a different appearance than those described. The user can change the cursor appearance by adjusting the mouse pointer settings in their operating system. The panning and no move cursors are static and cannot be changed by the operating system.

**b) AppStarting**

**c) ToString**

[C#] public static Cursor AppStarting {get;}

[C++] public: \_\_property static Cursor\* get\_AppStarting();

[VB] Public Shared ReadOnly Property AppStarting As Cursor

[JScript] public static function get AppStarting() : Cursor;

#### *Description*

Gets the cursor that appears when an application starts.

**d) Arrow**

**e) ToString**

[C#] public static Cursor Arrow {get;}

[C++] public: \_\_property static Cursor\* get\_Arrow();

[VB] Public Shared ReadOnly Property Arrow As Cursor

[JScript] public static function get Arrow() : Cursor;

#### *Description*

Gets the arrow cursor.

**f) Cross**

**g) ToString**

[C#] public static Cursor Cross {get;}

[C++] public: \_\_property static Cursor\* get\_Cross();

1 [VB] Public Shared ReadOnly Property Cross As Cursor

2 [JScript] public static function get Cross() : Cursor;

3  
4 *Description*

5 Gets the crosshair cursor.

6 *h) Default*

7 *i) ToString*

8  
9 [C#] public static Cursor Default {get;}

10 [C++] public: \_\_property static Cursor\* get\_Default();

11 [VB] Public Shared ReadOnly Property Default As Cursor

12 [JScript] public static function get Default() : Cursor;

13  
14 *Description*

15 Gets the default cursor, which is usually an arrow cursor.

16 *j) Hand*

17 *k) ToString*

18  
19 [C#] public static Cursor Hand {get;}

20 [C++] public: \_\_property static Cursor\* get\_Hand();

21 [VB] Public Shared ReadOnly Property Hand As Cursor

22 [JScript] public static function get Hand() : Cursor;

23  
24 *Description*

25 Gets the hand cursor, typically used when hovering over a Web link.



1) *Help*

m) *ToString*

[C#] public static Cursor Help {get;}

[C++] public: \_\_property static Cursor\* get\_Help();

[VB] Public Shared ReadOnly Property Help As Cursor

[JScript] public static function get Help() : Cursor;

*Description*

Gets the Help cursor, which is a combination of an arrow and a question mark.

n) *HSplit*

o) *ToString*

[C#] public static Cursor HSplit {get;}

[C++] public: \_\_property static Cursor\* get\_HSplit();

[VB] Public Shared ReadOnly Property HSplit As Cursor

[JScript] public static function get HSplit() : Cursor;

*Description*

Gets the cursor that appears when the mouse is positioned over a horizontal splitter bar.

p) *IBeam*

q) *ToString*

[C#] public static Cursor IBeam {get;}

```

1  [C++]    public:    __property    static    Cursor*    get_IBeam();
2  [VB]    Public    Shared    ReadOnly    Property    IBeam    As    Cursor
3  [JScript]    public    static    function    get    IBeam()    :    Cursor;

```

#### *Description*

Gets the I-beam cursor, which is used to show where the text cursor appears when the mouse is clicked.

*r) No*

*s) ToString*

```

10 [C#]      public      static      Cursor      No      {get;}
11 [C++]    public:    __property    static    Cursor*    get_No();
12 [VB]    Public    Shared    ReadOnly    Property    No    As    Cursor
13 [JScript]    public    static    function    get    No()    :    Cursor;

```

#### *Description*

Gets the cursor that indicates that a particular region is invalid for the current operation.

*t) NoMove2D*

*u) ToString*

```

21 [C#]      public      static      Cursor      NoMove2D      {get;}
22 [C++]    public:    __property    static    Cursor*    get_NoMove2D();
23 [VB]    Public    Shared    ReadOnly    Property    NoMove2D    As    Cursor
24 [JScript]    public    static    function    get    NoMove2D()    :    Cursor;

```

## Description

Gets the cursor that appears during wheel operations when the mouse is not moving, but the window can be scrolled in both a horizontal and vertical direction.

v) *NoMoveHoriz*

w) *ToString*

```
[C#]      public      static      Cursor      NoMoveHoriz      {get;}
```

```
[C++]     public:     __property     static     Cursor*     get_NoMoveHoriz();
```

```
[VB]      Public      Shared      ReadOnly      Property      NoMoveHoriz      As      Cursor
```

```
[JScript] public      static      function      get      NoMoveHoriz()      :      Cursor;
```

## Description

Gets the cursor that appears during wheel operations when the mouse is not moving, but the window can be scrolled in a horizontal direction.

x) *NoMoveVert*

y) *ToString*

```
[C#]      public      static      Cursor      NoMoveVert      {get;}
```

```
[C++]     public:     __property     static     Cursor*     get_NoMoveVert();
```

```
[VB]      Public      Shared      ReadOnly      Property      NoMoveVert      As      Cursor
```

```
[JScript] public      static      function      get      NoMoveVert()      :      Cursor;
```

## Description

Gets the cursor that appears during wheel operations when the mouse is not moving, but the window can be scrolled in a vertical direction.

*z) PanEast*

*aa) ToString*

```
[C#]      public      static      Cursor      PanEast      {get;}
```

```
[C++]      public:      __property      static      Cursor*      get_PanEast();
```

```
[VB]      Public      Shared      ReadOnly      Property      PanEast      As      Cursor
```

```
[JScript]      public      static      function      get      PanEast()      :      Cursor;
```

#### *Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally to the right.

*bb) PanNE*

*cc) ToString*

```
[C#]      public      static      Cursor      PanNE      {get;}
```

```
[C++]      public:      __property      static      Cursor*      get_PanNE();
```

```
[VB]      Public      Shared      ReadOnly      Property      PanNE      As      Cursor
```

```
[JScript]      public      static      function      get      PanNE()      :      Cursor;
```

#### *Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally and vertically upward and to the right.

**dd) PanNorth**

**ee) ToString**

```
[C#]      public      static      Cursor      PanNorth      {get;}
[C++]     public:     __property static      Cursor*      get_PanNorth();
[VB]      Public     Shared     ReadOnly     Property     PanNorth     As     Cursor
[JScript] public      static      function     get      PanNorth()      :      Cursor;
```

*Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling vertically in an upward direction.

**ff) PanNW**

**gg) ToString**

```
[C#]      public      static      Cursor      PanNW      {get;}
[C++]     public:     __property static      Cursor*      get_PanNW();
[VB]      Public     Shared     ReadOnly     Property     PanNW     As     Cursor
[JScript] public      static      function     get      PanNW()      :      Cursor;
```

*Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally and vertically upward and to the left.

**hh) PanSE**

**ii) ToString**

```
[C#]      public      static      Cursor      PanSE      {get;}
```

```

1  [C++]      public:      __property      static      Cursor*      get_PanSE();
2  [VB]      Public      Shared      ReadOnly      Property      PanSE      As      Cursor
3  [JScript]  public      static      function      get      PanSE()      :      Cursor;

```

#### *Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally and vertically downward and to the right.

*jj) PanSouth*

*kk) ToString*

```

10 [C#]      public      static      Cursor      PanSouth      {get;}
11
12 [C++]      public:      __property      static      Cursor*      get_PanSouth();
13 [VB]      Public      Shared      ReadOnly      Property      PanSouth      As      Cursor
14 [JScript]  public      static      function      get      PanSouth()      :      Cursor;

```

#### *Description*

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling vertically in a downward direction.

*ll) PanSW*

*mm) ToString*

```

21 [C#]      public      static      Cursor      PanSW      {get;}
22
23 [C++]      public:      __property      static      Cursor*      get_PanSW();
24 [VB]      Public      Shared      ReadOnly      Property      PanSW      As      Cursor
25 [JScript]  public      static      function      get      PanSW()      :      Cursor;

```

## Description

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally and vertically downward and to the left.

**nn) PanWest**

**oo) ToString**

```
[C#]      public      static      Cursor      PanWest      {get;}
[C++]     public:     __property static      Cursor*      get_PanWest();
[VB]      Public     Shared     ReadOnly     Property     PanWest     As     Cursor
[JScript] public      static      function     get      PanWest()      :      Cursor;
```

## Description

Gets the cursor that appears during wheel operations when the mouse is moving and the window is scrolling horizontally to the left.

**pp) SizeAll**

**qq) ToString**

```
[C#]      public      static      Cursor      SizeAll      {get;}
[C++]     public:     __property static      Cursor*      get_SizeAll();
[VB]      Public     Shared     ReadOnly     Property     SizeAll     As     Cursor
[JScript] public      static      function     get      SizeAll()      :      Cursor;
```

## Description

Gets the four-headed sizing cursor, which consists of four joined arrows that point north, south, east, and west.

**rr) SizeNESW**

**ss) ToString**

[C#] public static Cursor SizeNESW {get;}

[C++] public: \_\_property static Cursor\* get\_SizeNESW();

[VB] Public Shared ReadOnly Property SizeNESW As Cursor

[JScript] public static function get SizeNESW() : Cursor;

#### *Description*

Gets the two-headed diagonal (northeast/southwest) sizing cursor.

**tt) SizeNS**

**uu) ToString**

[C#] public static Cursor SizeNS {get;}

[C++] public: \_\_property static Cursor\* get\_SizeNS();

[VB] Public Shared ReadOnly Property SizeNS As Cursor

[JScript] public static function get SizeNS() : Cursor;

#### *Description*

Gets the two-headed vertical (north/south) sizing cursor.

**vv) SizeNWSE**

**ww) ToString**

[C#] public static Cursor SizeNWSE {get;}

[C++] public: \_\_property static Cursor\* get\_SizeNWSE();



1 [VB] Public Shared ReadOnly Property SizeNWSE As Cursor

2 [JScript] public static function get SizeNWSE() : Cursor;

3  
4 *Description*

5 Gets the two-headed diagonal (northwest/southeast) sizing cursor.

6 **xx) SizeWE**

7 **yy) ToString**

8  
9 [C#] public static Cursor SizeWE {get;}

10 [C++] public: \_\_property static Cursor\* get\_SizeWE();

11 [VB] Public Shared ReadOnly Property SizeWE As Cursor

12 [JScript] public static function get SizeWE() : Cursor;

13  
14 *Description*

15 Gets the two-headed horizontal (west/east) sizing cursor.

16 **zz) UpArrow**

17 **aaa) ToString**

18  
19 [C#] public static Cursor UpArrow {get;}

20 [C++] public: \_\_property static Cursor\* get\_UpArrow();

21 [VB] Public Shared ReadOnly Property UpArrow As Cursor

22 [JScript] public static function get UpArrow() : Cursor;

23  
24 *Description*

25 Gets the up arrow cursor, typically used to identify an insertion point.

**bbb) VSplit**

**ccc) ToString**

```
[C#]      public      static      Cursor      VSplit      {get;}
[C++]     public:     __property     static     Cursor*     get_VSplit();
[VB]      Public      Shared      ReadOnly      Property      VSplit      As      Cursor
[JScript] public      static      function     get      VSplit()      :      Cursor;
```

*Description*

Gets the cursor that appears when the mouse is positioned over a vertical splitter bar.

**ddd) WaitCursor**

**eee) ToString**

```
[C#]      public      static      Cursor      WaitCursor      {get;}
[C++]     public:     __property     static     Cursor*     get_WaitCursor();
[VB]      Public      Shared      ReadOnly      Property      WaitCursor      As      Cursor
[JScript] public      static      function     get      WaitCursor()      :      Cursor;
```

*Description*

Gets the wait cursor, typically an hourglass shape.

DataFormats class (System.Windows.Forms)

*a) ToString*

*Description*

Provides **static** , predefined **System.Windows.Forms.Clipboard** format names. Use them to identify the format of data that you store in an **System.Windows.Forms.IDataObject** .

The **System.Windows.Forms.IDataObject** and **System.Windows.Forms.DataObject** classes also use the **static** format list to determine the type of data that is retrieved from the system **System.Windows.Forms.Clipboard** , or that is transferred in a drag-and-drop operation.

*b) ToString*

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Bitmap;   |
| [C++]     | public: | static |          | String* | Bitmap;   |
| [VB]      | Public  | Shared | ReadOnly | Bitmap  | As String |
| [JScript] | public  | static | var      | Bitmap  | : String; |

*Description*

Specifies a Windows bitmap format. This **static** field is read-only.

A bitmap represents a computer graphic as an array of bits in memory, and these bits represent the attributes of the individual pixels in an image.

*c) ToString*

|       |         |        |          |         |                      |
|-------|---------|--------|----------|---------|----------------------|
| [C#]  | public  | static | readonly | string  | CommaSeparatedValue; |
| [C++] | public: | static |          | String* | CommaSeparatedValue; |

[VB] Public Shared ReadOnly CommaSeparatedValue As String

[JScript] public static var CommaSeparatedValue : String;

#### *Description*

Specifies a comma-separated value (CSV) format, which is a common interchange format used by spreadsheets. This format is not used directly by Windows Forms. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

#### *d) ToString*

[C#] public static readonly string Dib;

[C++] public: static String\* Dib;

[VB] Public Shared ReadOnly Dib As String

[JScript] public static var Dib : String;

#### *Description*

Specifies the Windows Device Independent Bitmap (DIB) format. This **static** field is read-only.

DIB is a file format designed to ensure that bitmapped graphics created using one application can be loaded and displayed in another application exactly the way they appeared in the originating application.

#### *e) ToString*

[C#] public static readonly string Dif;

[C++] public: static String\* Dif;

[VB] Public Shared ReadOnly Dif As String

```
[JScript]      public      static      var      Dif      :      String;
```

### *Description*

Specifies the Windows Data Interchange Format (DIF), which Windows Forms does not directly use. This **static** field is read-only.

DIF is a format consisting of ASCII codes in which database, spreadsheet, and similar documents can be structured to facilitate their use by and transfer to other programs.

### *f) ToString*

```
[C#]      public      static      readonly      string      EnhancedMetafile;
```

```
[C++]      public:      static      String*      EnhancedMetafile;
```

```
[VB]      Public      Shared      ReadOnly      EnhancedMetafile      As      String
```

```
[JScript]      public      static      var      EnhancedMetafile      :      String;
```

### *Description*

Specifies the Windows enhanced metafile format. This **static** field is read-only.

The metafile format is a Windows file that stores an image in terms of graphic objects rather than pixels. When resized, a metafile preserves an image better than a bitmap.

### *g) ToString*

```
[C#]      public      static      readonly      string      FileDrop;
```

```
[C++]      public:      static      String*      FileDrop;
```

```
[VB]      Public      Shared      ReadOnly      FileDrop      As      String
```

```
[JScript]      public      static      var      FileDrop      :      String;
```

### Description

Specifies the Windows file drop format, which Windows Forms does not directly use. This **static** field is read-only.

You can use this format to interact with shell file drags during drag-and-drop operations.

#### h) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Html;     |
| [C++]     | public: | static |          | String* | Html;     |
| [VB]      | Public  | Shared | ReadOnly | Html    | As String |
| [JScript] | public  | static | var      | Html    | : String; |

### Description

Specifies text consisting of HTML data. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

#### i) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Locale;   |
| [C++]     | public: | static |          | String* | Locale;   |
| [VB]      | Public  | Shared | ReadOnly | Locale  | As String |
| [JScript] | public  | static | var      | Locale  | : String; |

### Description

Specifies the Windows culture format, which Windows Forms does not directly use. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

j) *ToString*

|           |         |        |          |              |               |
|-----------|---------|--------|----------|--------------|---------------|
| [C#]      | public  | static | readonly | string       | MetafilePict; |
| [C++]     | public: | static |          | String*      | MetafilePict; |
| [VB]      | Public  | Shared | ReadOnly | MetafilePict | As String     |
| [JScript] | public  | static | var      | MetafilePict | : String;     |

*Description*

Specifies the Windows metafile format, which Windows Forms does not directly use. This **static** field is read-only.

The metafile format is a Windows file that stores an image in terms of graphic objects rather than pixels. When resized, a metafile preserves an image better than a bitmap.

k) *ToString*

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | OemText;  |
| [C++]     | public: | static |          | String* | OemText;  |
| [VB]      | Public  | Shared | ReadOnly | OemText | As String |
| [JScript] | public  | static | var      | OemText | : String; |

*Description*

Specifies the standard Windows original equipment manufacturer (OEM) text format. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

### l) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Palette;  |
| [C++]     | public: | static |          | String* | Palette;  |
| [VB]      | Public  | Shared | ReadOnly | Palette | As String |
| [JScript] | public  | static | var      | Palette | : String; |

#### Description

Specifies the Windows palette format. This **static** field is read-only.

Use this field to query a **System.Windows.Forms.DataObject** for the format of data that it contains.

### m) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | PenData;  |
| [C++]     | public: | static |          | String* | PenData;  |
| [VB]      | Public  | Shared | ReadOnly | PenData | As String |
| [JScript] | public  | static | var      | PenData | : String; |

#### Description

Specifies the Windows pen data format, which consists of pen strokes for handwriting software; Windows Forms does not use this format. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

### n) ToString

|      |        |        |          |        |       |
|------|--------|--------|----------|--------|-------|
| [C#] | public | static | readonly | string | Riff; |
|------|--------|--------|----------|--------|-------|



|           |         |        |          |                |
|-----------|---------|--------|----------|----------------|
| [C++]     | public: | static | String*  | Riff;          |
| [VB]      | Public  | Shared | ReadOnly | Riff As String |
| [JScript] | public  | static | var      | Riff : String; |

#### *Description*

Specifies the Resource Interchange File Format (RIFF) audio format, which Windows Forms does not directly use. This **static** field is read-only.

RIFF is a broad-based specification, designed to be used in defining standard formats for different types of multimedia files.

#### *o) ToString*

|           |         |        |          |               |      |
|-----------|---------|--------|----------|---------------|------|
| [C#]      | public  | static | readonly | string        | Rtf; |
| [C++]     | public: | static | String*  | Rtf;          |      |
| [VB]      | Public  | Shared | ReadOnly | Rtf As String |      |
| [JScript] | public  | static | var      | Rtf : String; |      |

#### *Description*

Specifies text consisting of Rich Text Format (RTF) data. This **static** field is read-only.

RTF is an adaptation of Document Content Architecture that is used for transferring formatted text documents between applications.

#### *p) ToString*

|       |         |        |          |                        |               |
|-------|---------|--------|----------|------------------------|---------------|
| [C#]  | public  | static | readonly | string                 | Serializable; |
| [C++] | public: | static | String*  | Serializable;          |               |
| [VB]  | Public  | Shared | ReadOnly | Serializable As String |               |

[JScript] public static var Serializable : String;

### *Description*

Specifies a format that encapsulates any type of Windows Forms object. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

#### *q) ToString*

[C#] public static readonly string StringFormat;

[C++] public: static String\* StringFormat;

[VB] Public Shared ReadOnly StringFormat As String

[JScript] public static var StringFormat : String;

### *Description*

Specifies the Windows Forms string class format, which Windows Forms uses to store string objects. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

#### *r) ToString*

[C#] public static readonly string SymbolicLink;

[C++] public: static String\* SymbolicLink;

[VB] Public Shared ReadOnly SymbolicLink As String

[JScript] public static var SymbolicLink : String;

### Description

Specifies the Windows symbolic link format, which Windows Forms does not directly use. This **static** field is read-only.

A symbolic link is a disk directory entry that takes the place of a directory entry for a file, but is actually a reference to a file in a different directory. A symbolic link is also called an alias, shortcut, soft link, or symlink.

#### s) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Text;     |
| [C++]     | public: | static |          | String* | Text;     |
| [VB]      | Public  | Shared | ReadOnly | Text    | As String |
| [JScript] | public  | static | var      | Text    | : String; |

### Description

Specifies the standard ANSI text format. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

#### t) ToString

|           |         |        |          |         |           |
|-----------|---------|--------|----------|---------|-----------|
| [C#]      | public  | static | readonly | string  | Tiff;     |
| [C++]     | public: | static |          | String* | Tiff;     |
| [VB]      | Public  | Shared | ReadOnly | Tiff    | As String |
| [JScript] | public  | static | var      | Tiff    | : String; |

### Description

Specifies the Tagged Image File Format (TIFF), which Windows Forms does not directly use. This **static** field is read-only.

TIFF is a standard file format commonly used for scanning, storage, and interchanges of gray-scale graphic images.

*u) ToString*

[C#]        public        static        readonly        string        UnicodeText;

[C++]        public:        static        String\*        UnicodeText;

[VB]        Public        Shared        ReadOnly        UnicodeText        As        String

[JScript]        public        static        var        UnicodeText        :        String;

*Description*

Specifies the standard Windows Unicode text format. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

*v) ToString*

[C#]        public        static        readonly        string        WaveAudio;

[C++]        public:        static        String\*        WaveAudio;

[VB]        Public        Shared        ReadOnly        WaveAudio        As        String

[JScript]        public        static        var        WaveAudio        :        String;

*Description*

Specifies the wave audio format, which Windows Forms does not directly use. This **static** field is read-only.

This field is used by the **System.Windows.Forms.IDataObject** interface and the **System.Windows.Forms.DataObject** class to specify the data type.

w) *GetFormat*

```
[C#]      public      static      Format      GetFormat(int      id);
[C++]     public:     static      Format*      GetFormat(int      id);
[VB] Public Shared Function GetFormat(ByVal id As Integer) As Format
[JScript] public static function GetFormat(id : int) : Format;
```

*Description*

Returns a **System.Windows.Forms.DataFormats.Format** with the Windows Clipboard numeric ID and name for the specified ID.

*Return Value:* A **System.Windows.Forms.DataFormats.Format** that has the Windows Clipboard numeric ID and the name of the format.

This member is typically used to register native clipboard formats. The format ID.

x) *GetFormat*

```
[C#]      public      static      Format      GetFormat(string      format);
[C++]     public:     static      Format*      GetFormat(String*      format);
[VB] Public Shared Function GetFormat(ByVal format As String) As Format
[JScript] public static function GetFormat(format : String) : Format; Returns a
System.Windows.Forms.DataFormats.Format with the Windows Clipboard
numeric ID and name.
```

*Description*

Returns a **System.Windows.Forms.DataFormats.Format** with the Windows Clipboard numeric ID and name for the specified format.

*Return Value:* A **System.Windows.Forms.DataFormats.Format** that has the Windows Clipboard numeric ID and the name of the format.

Call **System.Windows.Forms.DataFormats.GetFormat(System.String)** with the format name when you need a Windows Clipboard numeric ID for an existing format. The format name.

DataGrid class (System.Windows.Forms)

*a) ToString*

*Description*

Displays ADO.NET data in a scrollable grid.

The **System.Windows.Forms.DataGrid** displays web-like links to child tables. You can click on a link to navigate to the child table. When a child table is displayed, a back button appears in the caption that can be clicked to navigate back to the parent table. The data from the parent rows is displayed below the caption and above the column headers. You can hide the parent row information by clicking the button to the right of the back button.

*b) ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | int            | AutoColumnSize; |
| [C++]     | public: | const | int            | AutoColumnSize; |
| [VB]      | Public  | Const | AutoColumnSize | As Integer      |
| [JScript] | public  | var   | AutoColumnSize | : int;          |

*Description*

Specifies that the grid automatically sizes columns to the maximum width of the first ten rows. This field is constant.

Set **System.Windows.Forms.DataGrid.PreferredColumnWidth** to **System.Windows.Forms.DataGrid.AutoColumnSize** to automatically size grid columns.

c) ***DataGrid***

*Example Syntax:*

d) ***ToString***

[C#] public DataGrid();

[C++] public: DataGrid();

[VB] Public Sub New()

[JScript] public function DataGrid();

*Description*

Initializes a new instance of the **System.Windows.Forms.DataGrid** class.

To populate a newly created **System.Windows.Forms.DataGrid** control, set the **System.Windows.Forms.DataGrid.DataSource** property to a valid source, such as a **System.Data.DataView** , **System.Data.DataSet** , or **System.Data.DataViewManager** .

e) ***AccessibilityObject***

f) ***AccessibleDefaultActionDescription***

g) ***AccessibleDescription***

h) ***AccessibleName***

i) ***AccessibleRole***

j) ***AllowDrop***

k) ***AllowNavigation***

l) ***ToString***

*Description*

Gets or sets a value indicating whether navigation is allowed.

If this property is set to **false** , no links to child tables are shown.

*m) AllowSorting*

*n) ToString*

```
[C#]      public      bool      AllowSorting      {get;      set;}
```

```
[C++] public: __property bool get_AllowSorting();public: __property void  
set_AllowSorting(bool);
```

```
[VB]      Public      Property      AllowSorting      As      Boolean
```

```
[JScript] public function get AllowSorting() : Boolean;public function set  
AllowSorting(Boolean);
```

### *Description*

Gets or sets a value indicating whether the grid can be resorted by clicking on a column header.

You can also sort using an expression for a **System.Data.DataColumn** . See **System.Data.DataColumn.Expression** for details on creating a sort expression.

*o) AlternatingBackColor*

*p) ToString*

```
[C#]      public      Color      AlternatingBackColor      {get;      set;}
```

```
[C++] public: __property Color get_AlternatingBackColor();public: __property  
void set_AlternatingBackColor(Color);
```

```
[VB]      Public      Property      AlternatingBackColor      As      Color
```

```
[JScript] public function get AlternatingBackColor() : Color;public function set
```



AlternatingBackColor(Color);

### Description

Gets or sets the background color of alternating rows for a ledger appearance.

By default, all rows have the same color (the **System.Windows.Forms.Control.BackColor** property of the control). When you set the **System.Windows.Forms.DataGrid.AlternatingBackColor** to a new color, every other row is set to the new color.

q) *Anchor*

r) *BackColor*

s) *ToString*

### Description

Gets or sets the background color of the grid.

Whereas the **System.Windows.Forms.DataGrid.BackColor** property determines the color of rows in the grid, the **System.Windows.Forms.DataGrid.BackgroundColor** determines the color of the nonrow area, which is only visible when the grid is scrolled to the bottom, or if only a few rows are contained in the grid.

t) *BackgroundColor*

u) *ToString*

[C#]        public        Color        BackgroundColor        {get;        set;}

[C++] public: \_\_property Color get\_BackgroundColor();public: \_\_property void  
set\_BackgroundColor(Color);

[VB]        Public        Property        BackgroundColor        As        Color

[JScript] public function get BackgroundColor() : Color;public function set BackgroundColor(Color);

#### *Description*

Gets or sets the color of the nonrow area of the grid.

The **System.Windows.Forms.DataGrid.BackgroundColor** determines the color of the nonrow area of the grid, which is only visible when no table is displayed by the **System.Windows.Forms.DataGrid** , or if the grid is scrolled to the bottom, or if only a few rows are contained in the grid.

v) *BackgroundImage*

w) *ToString*

[C#] public override Image BackgroundImage {get; set;}

[C++] public: \_\_property virtual Image\* get\_BackgroundImage();public: \_\_property virtual void set\_BackgroundImage(Image\*);

[VB] Overrides Public Property BackgroundImage As Image

[JScript] public function get BackgroundImage() : Image;public function set BackgroundImage(Image);

#### *Description*

Gets or sets the background image for the control.

x) *BindingContext*

y) *BorderStyle*

z) *ToString*

*Description*

Gets or sets the grid's border style.

aa) *Bottom*

bb) *Bounds*

cc) *CanFocus*

dd) *CanSelect*

ee) *CaptionBackColor*

ff) *ToString*

*Description*

Gets or sets the background color of the caption area.

gg) *CaptionFont*

hh) *ToString*

[C#]            public            Font            CaptionFont            {get;            set;}

[C++]   public: \_\_property Font\*   get\_CaptionFont();public: \_\_property void  
set\_CaptionFont(Font\*);

[VB]            Public            Property            CaptionFont            As            Font

[JScript] public function get CaptionFont() : Font;public function set  
CaptionFont(Font);

*Description*

Gets or sets the font of the grid's caption.

**A System.Drawing.Font** encapsulates a Windows font and provides the methods for manipulating that font.

*ii) CaptionForeColor*

*jj) ToString*

[C#] public Color CaptionForeColor {get; set;}

[C++] public: \_\_property Color get\_CaptionForeColor();public: \_\_property void  
set\_CaptionForeColor(Color);

[VB] Public Property CaptionForeColor As Color

[JScript] public function get CaptionForeColor() : Color;public function set  
CaptionForeColor(Color);

*Description*

Gets or sets the foreground color of the caption area.

*kk) CaptionText*

*ll) ToString*

[C#] public string CaptionText {get; set;}

[C++] public: \_\_property String\* get\_CaptionText();public: \_\_property void  
set\_CaptionText(String\*);

```

1  [VB]      Public      Property      CaptionText      As      String
2  [JScript] public function get CaptionText() : String;public function set
3  CaptionText(String);

```

#### *Description*

Gets or sets the text of the grid's window caption.

*mm) CaptionVisible*

*nn) ToString*

```

10 [C#]      public      bool      CaptionVisible      {get;      set;}

```

```

11 [C++] public: __property bool get_CaptionVisible();public: __property void
12 set_CaptionVisible(bool);

```

```

13 [VB]      Public      Property      CaptionVisible      As      Boolean

```

```

14 [JScript] public function get CaptionVisible() : Boolean;public function set
15 CaptionVisible(Boolean);

```

#### *Description*

Gets or sets a value that indicates whether the grid's caption is visible.

If **System.Windows.Forms.DataGrid.CaptionVisible** is **false**, the **Back** button, **ParentRow** button, and caption will not be seen. Because navigation is limited, links to child tables will also not be visible and **System.Windows.Forms.DataGrid.AllowNavigation** will be set to **None**. The following example toggles the **System.Windows.Forms.DataGrid.CaptionVisible** property.

- oo) Capture**
- pp) CausesValidation**
- qq) ClientRectangle**
- rr) ClientSize**
- ss) ColumnHeadersVisible**
- tt) ToString**

#### *Description*

Gets or sets a value indicating whether the parent rows of a table are visible.

- uu) CompanyName**
- vv) Container**
- ww) ContainsFocus**
- xx) ContextMenu**
- yy) Controls**
- zz) Created**
- aaa) CreateParams**
- bbb) CurrentCell**
- ccc) ToString**

#### *Description*

Gets or sets which cell has the focus. Not available at design time.

Setting the **System.Windows.Forms.DataGrid.CurrentRow** property will cause the grid to scroll and show the cell if it is not already visible.

*ddd) CurrentRowIndex*

*eee) ToString*

[C#]        public        int        CurrentRowIndex        {get;        set;}

[C++] public: \_\_property int get\_CurrentRowIndex();public: \_\_property void  
set\_CurrentRowIndex(int);

[VB]        Public        Property        CurrentRowIndex        As        Integer

[JScript] public function get CurrentRowIndex() : int;public function set  
CurrentRowIndex(int);

### *Description*

Gets or sets index of the selected row.

The **System.Windows.Forms.DataGrid.CurrentRow** property allows you to iterate through a parent table's rows even if you are viewing the child table rows. For example, if you are viewing a child table, incrementing the **System.Windows.Forms.DataGrid.CurrentRow** will cause the **System.Windows.Forms.DataGrid** to display the next set of records in the child table that are linked to the parent table.

*fff) Cursor*

*ggg) ToString*

[C#]        public        override        Cursor        Cursor        {get;        set;}

[C++] public: \_\_property virtual Cursor\* get\_Cursor();public: \_\_property virtual  
void        set\_Cursor(Cursor\*);

[VB]        Overrides        Public        Property        Cursor        As        Cursor

[JScript] public function get Cursor() : Cursor;public function set Cursor(Cursor);

### *Description*

Gets or sets the cursor for the control.

*hhh) DataBindings*

*iii) DataMember*

*jjj) ToString*

### *Description*

Gets or sets the specific list in a **System.Windows.Forms.DataGrid.DataSource** for which the **System.Windows.Forms.DataGrid** control displays a grid.

If a **System.Windows.Forms.DataGrid.DataSource** contains multiple sources of data, you should set the **System.Windows.Forms.DataGrid.DataMember** to one of the sources. For example, if the **System.Windows.Forms.DataGrid.DataSource** is a **System.Data.DataSet** or **System.Data.DataViewManager** that contains three tables named Customers, Orders, and OrderDetails, you must specify one of the tables to bind to. If the **System.Data.DataSet** or **System.Data.DataViewManager** contains only one **System.Data.DataTable**, you should set the **System.Windows.Forms.DataGrid.DataMember** to the **System.Data.DataTable.TableName** of that **System.Data.DataTable**.

*kkk) DataSource*

*lll) ToString*

[C#]            public            object            DataSource            {get;            set;}

[C++] public: \_\_property Object\* get\_DataSource();public: \_\_property void set\_DataSource(Object\*);



```

1  [VB]      Public      Property      DataSource      As      Object
2  [JScript] public function get DataSource() : Object;public function set
3  DataSource(Object);
4

```

#### *Description*

Gets or sets the data source that the grid is displaying data for.

At run time, use the

**System.Windows.Forms.DataGrid.SetDataBinding(System.Object, System.String)** method to set the **System.Windows.Forms.DataGrid.DataSource** and **System.Windows.Forms.DataGrid.DataMember** properties.

*mmm) DefaultImeMode*

*nnn) DefaultSize*

*ooo) ToString*

#### *Description*

Gets the default size of the control.

*ppp) DesignMode*

*qqq) DisplayRectangle*

*rrr) Disposing*

*sss) Dock*

*ttt) Enabled*

*uuu) Events*

*vvv) FirstVisibleColumn*

*www) ToString*

### *Description*

Gets the index of the first visible column in a grid.

A column is considered visible even if it is partially concealed.

*xxx) FlatMode*

*yyy) ToString*

[C#]            public            bool            FlatMode            {get;            set;}

[C++]   public:   \_\_property   bool   get\_FlatMode();public:   \_\_property   void  
set\_FlatMode(bool);

[VB]            Public            Property            FlatMode            As            Boolean

[JScript]   public   function   get   FlatMode()   :   Boolean;public   function   set  
FlatMode(Boolean);

### *Description*

Gets or sets a value indicating whether the grid displays in flat mode.

*zzz) Focused*

*aaaa) Font*

*bbbb) ForeColor*

*cccc) ToString*

#### *Description*

Gets or sets the foreground color (typically the color of the text) property of the **System.Windows.Forms.DataGrid** control.

*dddd) GridLineColor*

*eeee) ToString*

[C#]            public            Color            GridLineColor            {get;            set;}

[C++] public: \_\_property Color get\_GridLineColor();public: \_\_property void  
set\_GridLineColor(Color);

[VB]            Public            Property            GridLineColor            As            Color

[JScript] public function get GridLineColor() : Color;public function set  
GridLineColor(Color);

#### *Description*

Gets or sets the color of the grid lines.

No grid line is displayed if the  
**System.Windows.Forms.DataGrid.GridLineStyle** property is set to  
**DataGridLineStyle.None** .

***ffff) GridLineStyle***

***gggg) ToString***

```
[C#]      public      DataGridLineStyle      GridLineStyle      {get;      set;}
[C++]    public:  __property  DataGridLineStyle  get_GridLineStyle();public:
__property          void          set_GridLineStyle(DataGridLineStyle);
[VB]      Public      Property      GridLineStyle      As      DataGridLineStyle
[JScript] public function get GridLineStyle() : DataGridLineStyle;public function
set          GridLineStyle(DataGridLineStyle);
```

***Description***

Gets or sets the line style of the grid.

***hhhh) Handle***

***iiii) HasChildren***

***jjjj) HeaderBackColor***

***kkkk) ToString***

***Description***

Gets or sets the background color of all row and column headers.

***llll) HeaderFont***

***mmmm)ToString***

```
[C#]      public      Font      HeaderFont      {get;      set;}
```

[C++] public: \_\_property Font\* get\_HeaderFont();public: \_\_property void set\_HeaderFont(Font\*);

[VB] Public Property HeaderFont As Font

[JScript] public function get HeaderFont() : Font;public function set HeaderFont(Font);

### *Description*

Gets or sets the font used for headers.

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*nnnn) HeaderForeColor*

*oooo) ToString*

[C#] public Color HeaderForeColor {get; set;}

[C++] public: \_\_property Color get\_HeaderForeColor();public: \_\_property void set\_HeaderForeColor(Color);

[VB] Public Property HeaderForeColor As Color

[JScript] public function get HeaderForeColor() : Color;public function set HeaderForeColor(Color);

### *Description*

Gets or sets the foreground color of headers.

The following example sets the foreground color of the grid's column headers.

1        *pppp) Height*

2        *qqqq) HorizScrollBar*

3        *rrrr) ToString*

4  
5  
6        *Description*

7        Gets the horizontal scrollbar for the grid.

8        *ssss) ImeMode*

9        *tttt) InvokeRequired*

10       *uuuu) IsAccessible*

11       *vvvv) IsDisposed*

12       *wwwwww)IsHandleCreated*

13       *xxxx) Item*

14       *yyyy) ToString*

15  
16  
17       *Description*

18       Gets or sets the value of a specified **System.Windows.Forms.DataGridCell** .

19       Setting this property changes the position of the **System.Data.DataView** to  
20       the specified row. A **System.Windows.Forms.DataGridCell** that represents a  
21       cell in the grid.

22       *zzzz) Item*

23       *aaaaaa) ToString*

24  
25       [C#]    public    object    this[int    rowIndex,    int    columnIndex]    {get;    set;}

[C++] public: \_\_property Object\* get\_Item(int rowIndex, int columnIndex); public: \_\_property void set\_Item(int rowIndex, int columnIndex, Object\*);

[VB] Public Default Property Item(ByVal rowIndex As Integer, ByVal columnIndex As Integer) As Object

[JScript] returnValue = DataGridObject.Item(rowIndex, columnIndex); DataGridObject.Item(rowIndex, columnIndex) = returnValue; Gets or sets the value of a specified cell.

### *Description*

Gets or sets the value of the cell at the specified the row and column.

Setting this property changes the position of the **System.Data.DataView** to the specified row. The zero-based index of the row containing the value. The zero-based index of the column containing the value.

*bbbbbb) Left*

*ccccc) LinkColor*

*dddddd) ToString*

### *Description*

Gets or sets the color of the text that you can click to navigate to a child table.

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*eeee) LinkHoverColor*

*ffff) ToString*

```
[C#]      public      Color      LinkHoverColor      {get;      set;}
[C++] public: __property Color get_LinkHoverColor();public: __property void
set_LinkHoverColor(Color);
[VB]      Public      Property      LinkHoverColor      As      Color
[JScript] public function get LinkHoverColor() : Color;public function set
LinkHoverColor(Color);
```

#### *Description*

Gets or sets the color a link changes to when the mouse pointer moves over it.

*gggg) ListManager*

*hhhh)ToString*

```
[C#]  protected  internal  CurrencyManager  ListManager  {get;  set;}
[C++] internal:  __property  CurrencyManager*  get_ListManager();internal:
__property          void          set_ListManager(CurrencyManager*);
[VB]  Protected  Friend  Property  ListManager  As  CurrencyManager
[JScript] package function get ListManager() : CurrencyManager;package
function          set          ListManager(CurrencyManager);
```

#### *Description*

Gets the **System.Windows.Forms.CurrencyManager** for this **System.Windows.Forms.DataGrid** control.



*iiii) Location*

*jjjj) Name*

*kkkk) Parent*

*llll) ParentRowsBackColor*

*mmmm) ToString*

*Description*

Gets or sets the background color of parent rows.

*nnnn)ParentRowsForeColor*

*oooo) ToString*

[C#]        public        Color        ParentRowsForeColor        {get;        set;}

[C++] public: \_\_property Color get\_ParentRowsForeColor();public: \_\_property

void        set\_ParentRowsForeColor(Color);

[VB]        Public        Property        ParentRowsForeColor        As        Color

[JScript] public function get ParentRowsForeColor() : Color;public function set

ParentRowsForeColor(Color);

*Description*

Gets or sets the foreground color of parent rows.

*ppppp) ParentRowsLabelStyle*

*qqqqq) ToString*

[C#] public DataGridParentRowsLabelStyle ParentRowsLabelStyle {get; set;}

[C++] public: \_\_property DataGridParentRowsLabelStyle

get\_ParentRowsLabelStyle();public: \_\_property void

set\_ParentRowsLabelStyle(DataGridParentRowsLabelStyle);

[VB] Public Property ParentRowsLabelStyle As DataGridParentRowsLabelStyle

[JScript] public function get ParentRowsLabelStyle() :

DataGridParentRowsLabelStyle;public function set

ParentRowsLabelStyle(DataGridParentRowsLabelStyle);

### *Description*

Gets or sets the way parent row labels are displayed.

*rrrrr) ParentRowsVisible*

*sssss) ToString*

[C#] public bool ParentRowsVisible {get; set;}

[C++] public: \_\_property bool get\_ParentRowsVisible();public: \_\_property void

set\_ParentRowsVisible(bool);

[VB] Public Property ParentRowsVisible As Boolean

[JScript] public function get ParentRowsVisible() : Boolean;public function set

ParentRowsVisible(Boolean);

### *Description*

Gets or sets a value indicating whether the parent rows of a table are visible.

*tttt) PreferredColumnWidth*

*uuuuu)ToString*

[C#] public int PreferredColumnWidth {get; set;}

[C++] public: \_\_property int get\_PreferredColumnWidth();public: \_\_property

void set\_PreferredColumnWidth(int);

[VB] Public Property PreferredColumnWidth As Integer

[JScript] public function get PreferredColumnWidth() : int;public function set

PreferredColumnWidth(int);

#### *Description*

Gets or sets the default width of the grid columns in pixels.

Set this property before resetting the

**System.Windows.Forms.DataGrid.DataSource** and

**System.Windows.Forms.DataGrid.DataMember** properties (either separately, or through the

**System.Windows.Forms.DataGrid.SetDataBinding(System.Object,System.String)** method), or the property will have no effect.

*vvvvv) PreferredRowHeight*

*wwwww)ToString*

[C#] public int PreferredRowHeight {get; set;}

[C++] public: \_\_property int get\_PreferredRowHeight();public: \_\_property void

set\_PreferredRowHeight(int);

[VB] Public Property PreferredRowHeight As Integer

[JScript] public function get PreferredRowHeight() : int;public function set

1 PreferredRowHeight(int);

3 *Description*

4 Gets or sets the preferred row height for the  
5 **System.Windows.Forms.DataGrid** control.

6 Set this property before resetting the  
7 **System.Windows.Forms.DataGrid.DataSource** and  
8 **System.Windows.Forms.DataGrid.DataMember** properties (either  
9 separately, or through the  
10 **System.Windows.Forms.DataGrid.SetDataBinding(System.Object, System  
11 m.String)** method), or the property will have no effect.

12 *xxxxxx) ProductName*

13 *yyyyyy) ProductVersion*

14 *zzzzz) ReadOnly*

15 *aaaaaa) ToString*

16 *Description*

17 Gets or sets a value indicating whether the grid is in read-only mode.

18 In read-only mode, the grid can be scrolled, nodes can be expanded or  
19 collapsed, and so on, However, no additions, edits, or deletes may take place.

*bbbbbb)RecreatingHandle*

*cccccc)Region*

*dddddd)RenderRightToLeft*

*eeeeee)ResizeRedraw*

*ffffff) Right*

*gggggg)RightToLeft*

*hhhhh)RowHeadersVisible*

*iiii) ToString*

### *Description*

Gets or sets a value that specifies whether row headers are visible.

*jjjjj) RowHeaderWidth*

*kkkkkk)ToString*

[C#]        public        int        RowHeaderWidth        {get;        set;}

[C++] public: \_\_property int get\_RowHeaderWidth();public: \_\_property void  
set\_RowHeaderWidth(int);

[VB]        Public        Property        RowHeaderWidth        As        Integer

[JScript] public function get RowHeaderWidth() : int;public function set  
RowHeaderWidth(int);

### *Description*

Gets or sets the width of row headers.

#### *lllll) SelectionBackColor*

##### *mmmmm)ToString*

```
[C#]      public      Color      SelectionBackColor      {get;      set;}

[C++] public: __property Color get_SelectionBackColor();public: __property void
set_SelectionBackColor(Color);

[VB]      Public      Property      SelectionBackColor      As      Color

[JScript] public function get SelectionBackColor() : Color;public function set
SelectionBackColor(Color);
```

#### *Description*

Gets or sets the background color of selected rows.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid**.

##### *nnnnnn)SelectionForeColor*

##### *ooooo)ToString*

```
[C#]      public      Color      SelectionForeColor      {get;      set;}

[C++] public: __property Color get_SelectionForeColor();public: __property void
set_SelectionForeColor(Color);

[VB]      Public      Property      SelectionForeColor      As      Color

[JScript] public function get SelectionForeColor() : Color;public function set
SelectionForeColor(Color);
```

#### *Description*

Gets or set the foreground color of selected rows.

*pppppp)ShowFocusCues*

*qqqqqq)ShowKeyboardCues*

*rrrrrr) Site*

*ssssss) ToString*

*Description*

*tttttt) Size*

*uuuuuu)TabIndex*

*vvvvvv)TableStyles*

*wwwwww)ToString*

*Description*

Gets the collection of **System.Windows.Forms.DataGridTableStyle** objects for the grid.

Use the **System.Windows.Forms.GridTableStylesCollection** to create customized views of each table displayed by the **System.Windows.Forms.DataGrid** control.

xxxxxx)TabStop

yyyyyy)Tag

zzzzzz) Text

aaaaaaa)ToString

*Description*

Gets or sets text associated with the **System.Windows.Forms.DataGrid** control.

bbbbbbb)Top

ccccccc)TopLevelControl

ddddddd)VertScrollBar

eeeeeee)ToString

*Description*

Gets the vertical scroll bar of the control.

fffffff) Visible

ggggggg)VisibleColumnCount

hhhhhhh)ToString

*Description*

Gets the number of visible columns.



The number of visible columns can change depending on their width. For example, if a default width for all columns is set, but a new column's width is set twice as large, the number of visible columns will be reduced by at least one.

*iiiiii) VisibleRowCount*

*jjjjjj) ToString*

|           |         |            |                 |                            |
|-----------|---------|------------|-----------------|----------------------------|
| [C#]      | public  | int        | VisibleRowCount | {get;}                     |
| [C++]     | public: | __property | int             | get_VisibleRowCount();     |
| [VB]      | Public  | ReadOnly   | Property        | VisibleRowCount As Integer |
| [JScript] | public  | function   | get             | VisibleRowCount() : int;   |

#### *Description*

Gets the number of rows visible.

The number of visible rows can be changed at run time if the user is allowed to resize the **System.Windows.Forms.DataGrid** control.

*kkkkkkk)Width*

*llllll) WindowTarget*

*mmmmmmm)ToString*

#### *Description*

Occurs when the **System.Windows.Forms.DataGrid.AllowNavigation** property has changed.

If the **System.Windows.Forms.DataGrid.AllowNavigation** property is set to **false** , then no links to child tables are shown.

*nnnnnnnn)ToString*

|       |         |         |                 |                  |
|-------|---------|---------|-----------------|------------------|
| [C#]  | public  | event   | EventHandler    | BackButtonClick; |
| [C++] | public: | __event | EventHandler*   | BackButtonClick; |
| [VB]  | Public  | Event   | BackButtonClick | As EventHandler  |

*Description*

Occurs when the **Back** button on a child table is clicked.

The **Back** button becomes visible when a child table is displayed. Clicking the button will cause the grid to display the parent table.

*oooooooo)ToString*

*Description*

Occurs when the **System.Windows.Forms.DataGrid.BackgroundColor** has changed.

For more information about handling events, see .

*pppppppp)ToString*

*Description*

Occurs when the **System.Windows.Forms.DataGrid.BorderStyle** has changed.

Possible values include **None** , **FixedSingle** , and **Fixed3D** .

*qqqqqqq)ToString*

[C#]      public      event      EventHandler      CaptionVisibleChanged;  
[C++]      public:      \_\_event      EventHandler\*      CaptionVisibleChanged;  
[VB]      Public      Event      CaptionVisibleChanged      As      EventHandler

*Description*

Occurs when the **System.Windows.Forms.DataGrid.CaptionVisible** property has changed.

For more information about handling events, see .

*rrrrrrr)ToString*

*Description*

Occurs when the **System.Windows.Forms.DataGrid.CurrentCell** property has changed.

To determine the current cell, use the **System.Windows.Forms.DataGrid.CurrentCell** property.

*sssssss)ToString*

*Description*

Occurs when the **System.Windows.Forms.DataGrid.DataSource** property value has changed.

For more information about handling events, see .

1        *tttttt) ToString*

2  
3        *Description*

4        Occurs when the **System.Windows.Forms.DataGrid.FlatMode** has changed.  
5        For more information about handling events, see .

6        *uuuuuuu)ToString*

7  
8  
9        *Description*

10       Occurs when the user navigates to a new table.  
11       Use the **System.Windows.Forms.DataGrid.Navigate** event to reset  
12       individual column properties, such as Width, as appropriate to the table.

13       *vvvvvvv)ToString*

14  
15  
16       *Description*

17       Occurs when the label style of the parent row is changed.  
18       For more information about handling events, see .

19       *wwwwwww)ToString*

20  
21       [C#]       public       event       EventHandler       ParentRowsVisibleChanged;  
22       [C++]       public:       \_\_event       EventHandler\*       ParentRowsVisibleChanged;  
23       [VB]       Public       Event       ParentRowsVisibleChanged       As       EventHandler

24  
25       *Description*

Occurs when the **System.Windows.Forms.DataGrid.ParentRowsVisible** property value changes.

For more information about handling events, see .

*xxxxxxx)ToString*

#### *Description*

Occurs when the **System.Windows.Forms.DataGrid.ReadOnly** property value changes.

For more information about handling events, see .

*yyyyyyy)ToString*

#### *Description*

Occurs when a row header is clicked.

For more information about handling events, see .

*zzzzzzz)ToString*

|      |        |       |              |         |
|------|--------|-------|--------------|---------|
| [C#] | public | event | EventHandler | Scroll; |
|------|--------|-------|--------------|---------|

|       |         |         |               |         |
|-------|---------|---------|---------------|---------|
| [C++] | public: | __event | EventHandler* | Scroll; |
|-------|---------|---------|---------------|---------|

|      |        |       |        |    |              |
|------|--------|-------|--------|----|--------------|
| [VB] | Public | Event | Scroll | As | EventHandler |
|------|--------|-------|--------|----|--------------|

#### *Description*

Occurs when the user scrolls the **System.Windows.Forms.DataGrid** control.

### *aaaaaaaa)ToString*

```
1
2
3 [C#]    public    event    EventHandler    ShowParentDetailsButtonClick;
4 [C++]   public:   __event  EventHandler*    ShowParentDetailsButtonClick;
5 [VB]    Public    Event    ShowParentDetailsButtonClick    As    EventHandler
```

### *Description*

Occurs when the **ShowParentDetails** button is clicked.

### *bbbbbbbb)BeginEdit*

```
10 [C#] public bool BeginEdit(DataGridColumnStyle gridColumn, int rowNum);
11
12 [C++] public: __sealed bool BeginEdit(DataGridColumnStyle* gridColumn, int
13 rowNum);
14
15 [VB] NotOverridable Public Function BeginEdit(ByVal gridColumn As
16 DataGridColumnStyle, ByVal rowNum As Integer) As Boolean
17
18 [JScript] public function BeginEdit(gridColumn : DataGridColumnStyle,
19 rowNum : int) : Boolean;
```

### *Description*

Attempts to put the grid into a state where editing is allowed.

*Return Value:* **true** if the method is successful; otherwise, **false** . A

**System.Windows.Forms.DataGridColumnStyle** to edit. The number of the row to edit.

### *ccccccc)BeginInit*

```
24 [C#]                public                void                BeginInit();
25
```

|           |                |          |      |              |
|-----------|----------------|----------|------|--------------|
| [C++]     | public:        | __sealed | void | BeginInit(); |
| [VB]      | NotOverridable | Public   | Sub  | BeginInit()  |
| [JScript] | public         | function |      | BeginInit(); |

#### Description

Begins the initialization of a **System.Windows.Forms.DataGrid** that is used on a form or used by another component. The initialization occurs at run time.

The Visual Studio.NET design environment uses this method to start the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.DataGrid.EndInit** method ends the initialization. Using the **System.Windows.Forms.DataGrid.BeginInit** and **System.Windows.Forms.DataGrid.EndInit** methods prevents the control from being used before it is fully initialized.

#### ddddddd)CancelEditing

|           |             |           |      |                  |
|-----------|-------------|-----------|------|------------------|
| [C#]      | protected   | virtual   | void | CancelEditing(); |
| [C++]     | protected:  | virtual   | void | CancelEditing(); |
| [VB]      | Overridable | Protected | Sub  | CancelEditing()  |
| [JScript] | protected   | function  |      | CancelEditing(); |

#### Description

Cancels the current edit operation and rolls back all changes.

#### eeeeeeee)Collapse

|       |         |      |                |                 |
|-------|---------|------|----------------|-----------------|
| [C#]  | public  | void | Collapse(int   | row);           |
| [C++] | public: | void | Collapse(int   | row);           |
| [VB]  | Public  | Sub  | Collapse(ByVal | row As Integer) |

1 [JScript] public function Collapse(row : int);

3 *Description*

4 Collapses child relations, if any exist for all rows, or for a specified row.

5 Use the **System.Windows.Forms.DataGrid.IsExpanded(System.Int32)**  
6 method to determine if a row is expanded. The number of the row to collapse. If  
set to -1, all rows are collapsed.

7 *fffffff)ColumnStartedEditing*

9 [C#] protected internal virtual void ColumnStartedEditing(Control  
10 editingControl);

11 [C++] protected public: virtual void ColumnStartedEditing(Control\*  
12 editingControl);

13 [VB] Overridable Protected Friend Dim Sub ColumnStartedEditing(ByVal  
14 editingControl As Control)

15 [JScript] package function ColumnStartedEditing(editingControl : Control);

17 *Description*

18 Informs the **System.Windows.Forms.DataGrid** control when the user begins  
to edit a column using the specified control.

19 When called, the  
20 **System.Windows.Forms.IDataGridColumnStyleEditingNotificationService.ColumnStartedEditing(System.Windows.Forms.Control)** method  
21 allows the **System.Windows.Forms.DataGrid** control to show a pencil in the  
22 row header. The **System.Windows.Forms.Control** used to edit the column.

23 *gggggggg)ColumnStartedEditing*

25 [C#] protected internal virtual void ColumnStartedEditing(Rectangle bounds);



[C++] protected public: virtual void ColumnStartedEditing(Rectangle bounds);

[VB] Overridable Protected Friend Dim Sub ColumnStartedEditing(ByVal bounds  
As Rectangle)

[JScript] package function ColumnStartedEditing(bounds : Rectangle); Informs  
the **System.Windows.Forms.DataGrid** control that the user has begun editing a  
column.

### *Description*

Informs the **System.Windows.Forms.DataGrid** control when the user begins  
to edit the column at the specified location.

When called, the **System.Windows.Forms.IDataGridColumnStyleEditingNotificationService.ColumnStartedEditing(System.Windows.Forms.Control)** method  
allows the **System.Windows.Forms.DataGrid** control to show a pencil in the  
row header. The **System.Drawing.Rectangle** that defines the location of the  
edited column.

### *hhhhhhh)CreateAccessibilityInstance*

[C#] protected override AccessibleObject CreateAccessibilityInstance();

[C++] protected: AccessibleObject\* CreateAccessibilityInstance();

[VB] Overrides Protected Function CreateAccessibilityInstance() As  
AccessibleObject

[JScript] protected override function CreateAccessibilityInstance() :  
AccessibleObject;

### *Description*

Constructs a new instance of the accessibility object for this control.

*Return Value:* The **System.Windows.Forms.Control.ControlAccessibleObject** for this control.

Derived classes should not call the base class's  
**System.Windows.Forms.Control.CreateAccessibilityInstance** method.

*iiiiiii) CreateGridColumn*

```
[C#]           protected           virtual           DataGridColumnStyle
CreateGridColumn(PropertyDescriptor           prop);

[C++]           protected:           virtual           DataGridColumnStyle*
CreateGridColumn(PropertyDescriptor*           prop);

[VB] Overridable Protected Function CreateGridColumn(ByVal prop As
PropertyDescriptor)           As           DataGridColumnStyle

[JScript] protected function CreateGridColumn(prop : PropertyDescriptor) :
DataGridColumnStyle;
```

*Description*

Creates a new **System.Windows.Forms.DataGridColumnStyle** with the specified **System.ComponentModel.PropertyDescriptor** .  
*Return Value:* The new **System.Windows.Forms.DataGridColumnStyle** .  
The **System.ComponentModel.PropertyDescriptor** to use for creating the grid column style.

*jjjjjjj) CreateGridColumn*

```
[C#]           protected           virtual           DataGridColumnStyle
CreateGridColumn(PropertyDescriptor           prop,           bool           isDefault);

[C++]           protected:           virtual           DataGridColumnStyle*
CreateGridColumn(PropertyDescriptor*           prop,           bool           isDefault);

[VB] Overridable Protected Function CreateGridColumn(ByVal prop As
PropertyDescriptor, ByVal isDefault As Boolean) As DataGridColumnStyle
```

[JScript] protected function CreateGridColumn(prop : PropertyDescriptor,  
isDefault : Boolean) : DataGridColumnStyle; Creates a new  
**System.Windows.Forms.DataGridColumnStyle** that is added to the control.

#### *Description*

Creates a **System.Windows.Forms.DataGridColumnStyle** using the  
specified **System.ComponentModel.PropertyDescriptor**.

*Return Value:* The new **System.Windows.Forms.DataGridColumnStyle**.  
The **System.ComponentModel.PropertyDescriptor** to use for creating the  
grid column style. **true** to set the column style as the default; otherwise, **false**.

#### *kkkkkkkk)Dispose*

[C#] protected override void Dispose(bool disposing);

[C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

#### *Description*

Disposes of the resources (other than memory) used by the  
**System.Windows.Forms.DataGrid**.

Call **System.Windows.Forms.DataGrid.Dispose(System.Boolean)** when  
you are finished using the **System.Windows.Forms.DataGrid**. The  
**System.Windows.Forms.DataGrid.Dispose(System.Boolean)** method  
leaves the **System.Windows.Forms.DataGrid** in an unuseable state. After  
calling **System.Windows.Forms.DataGrid.Dispose(System.Boolean)**, you  
must release all references to the **System.Windows.Forms.DataGrid** so the  
memory it was occupying can be reclaimed by garbage collection.

#### *lllllll) EndEdit*

[C#] public bool EndEdit(DataGridColumnStyle gridColumn, int rowNumber,

```

1 bool shouldAbort);
2 [C++] public: __sealed bool EndEdit(DataGridColumnStyle* gridColumn, int
3 rowNumber, bool shouldAbort);
4 [VB] NotOverridable Public Function EndEdit(ByVal gridColumn As
5 DataGridColumnStyle, ByVal rowNumber As Integer, ByVal shouldAbort As
6 Boolean) As Boolean
7 [JScript] public function EndEdit(gridColumn : DataGridColumnStyle,
8 rowNumber : int, shouldAbort : Boolean) : Boolean;

```

#### Description

Requests an end to an edit operation taking place on the **System.Windows.Forms.DataGrid** control.

*Return Value:* **true** if the editing operation ceases; otherwise, **false**. The **System.Windows.Forms.DataGridColumnStyle** to cease editing. The number of the row to cease editing. Set to **true** if the current operation should be stopped.

#### *mmmmmmmm)EndInit*

```

16 [C#] public void EndInit();
17 [C++] public: __sealed void EndInit();
18 [VB] NotOverridable Public Sub EndInit()
19 [JScript] public function EndInit();

```

#### Description

Ends the initialization of a **System.Windows.Forms.DataGrid** that is used on a form or used by another component. The initialization occurs at run time.

The Visual Studio.NET design environment uses this method to end the initialization of a component that is used on a form or used by another component. The **System.Windows.Forms.DataGrid.BeginInit** method starts

the initialization. Using the **System.Windows.Forms.DataGrid.BeginInit** and **System.Windows.Forms.DataGrid.EndInit** methods prevents the control from being used before it is fully initialized.

### *nnnnnnnn)Expand*

|           |         |          |              |                 |
|-----------|---------|----------|--------------|-----------------|
| [C#]      | public  | void     | Expand(int   | row);           |
| [C++]     | public: | void     | Expand(int   | row);           |
| [VB]      | Public  | Sub      | Expand(ByVal | row As Integer) |
| [JScript] | public  | function | Expand(row   | : int);         |

### *Description*

Displays child relations, if any exist, for all rows or a specific row. The number of the row to expand. If set to -1, all rows are expanded.

### *oooooooo)GetCellBounds*

|           |         |           |                                          |              |
|-----------|---------|-----------|------------------------------------------|--------------|
| [C#]      | public  | Rectangle | GetCellBounds(DataGridCell               | dgc);        |
| [C++]     | public: | Rectangle | GetCellBounds(DataGridCell               | dgc);        |
| [VB]      | Public  | Function  | GetCellBounds(ByVal dgc As DataGridCell) | As Rectangle |
| [JScript] | public  | function  | GetCellBounds(dgc : DataGridCell)        | : Rectangle; |

### *Description*

Gets the **System.Drawing.Rectangle** of the cell specified by **System.Windows.Forms.DataGridCell**.

*Return Value:* A **System.Drawing.Rectangle** that defines the current cell's corners. The **System.Windows.Forms.DataGridCell** to look up.

### *ppppppppp)GetCellBounds*

```
1
2 [C#] public Rectangle GetCellBounds(int row, int col);
3 [C++] public: Rectangle GetCellBounds(int row, int col);
4 [VB] Public Function GetCellBounds(ByVal row As Integer, ByVal col As
5 Integer) As Rectangle
6 [JScript] public function GetCellBounds(row : int, col : int) : Rectangle; Gets the
7 System.Drawing.Rectangle that specifies the four corners of a cell.
8
```

#### *Description*

Gets the **System.Drawing.Rectangle** of the cell specified by row and column number.

*Return Value:* A **System.Drawing.Rectangle** that defines the current cell's corners. The number of the cell's row. The number of the cell's column.

### *qqqqqqqqq)GetCurrentCellBounds*

```
14
15 [C#] public Rectangle GetCurrentCellBounds();
16 [C++] public: Rectangle GetCurrentCellBounds();
17 [VB] Public Function GetCurrentCellBounds() As Rectangle
18 [JScript] public function GetCurrentCellBounds() : Rectangle;
19
```

#### *Description*

Gets a **System.Drawing.Rectangle** that specifies the four corners of the selected cell.

*Return Value:* A **System.Drawing.Rectangle** that defines the current cell's corners.

## *rrrrrrrr)GetOutputTextDelimiter*

```
1  
2 [C#]      protected      virtual      string      GetOutputTextDelimiter();  
3 [C++]     protected:     virtual      String*      GetOutputTextDelimiter();  
4 [VB]      Overridable Protected Function GetOutputTextDelimiter() As String  
5 [JScript] protected      function      GetOutputTextDelimiter() : String;  
6  
7
```

### *Description*

Will return the string that will be used as a delimiter between columns when copying rows contents to the Clipboard. At the moment, return "\t" Will return the string that will be used as a delimiter between columns when copying rows contents to the Clipboard. At the moment, return "\t"

## *ssssssss)GridHScrolled*

```
12  
13 [C#] protected virtual void GridHScrolled(object sender, ScrollEventArgs se);  
14 [C++] protected: virtual void GridHScrolled(Object* sender, ScrollEventArgs*  
15 se);  
16 [VB] Overridable Protected Sub GridHScrolled(ByVal sender As Object, ByVal  
17 se As ScrollEventArgs)  
18 [JScript] protected function GridHScrolled(sender : Object, se : ScrollEventArgs);  
19  
20
```

### *Description*

Listens for the horizontal scrollbar's scroll event. An **System.Object** that contains data about the control. A **System.Windows.Forms.ScrollEventArgs** that contains the event data.

### ttttttt) GridVScrolled

```
1
2 [C#] protected virtual void GridVScrolled(object sender, ScrollEventArgs se);
3
4 [C++] protected: virtual void GridVScrolled(Object* sender, ScrollEventArgs*
5 se);
6
7 [VB] Overridable Protected Sub GridVScrolled(ByVal sender As Object, ByVal
8 se
9 As ScrollEventArgs)
10
11 [JScript] protected function GridVScrolled(sender : Object, se : ScrollEventArgs);
12
```

#### Description

10 Listens for the vertical scrollbar's scroll event. An **System.Object** that contains  
11 data about the control. A **System.Windows.Forms.ScrollEventArgs** that  
12 contains the event data.

### uuuuuuuu)HitTest

```
13
14
15 [C#] public HitTestInfo HitTest(Point position);
16
17 [C++] public: HitTestInfo* HitTest(Point position);
18
19 [VB] Public Function HitTest(ByVal position As Point) As HitTestInfo
20
21 [JScript] public function HitTest(position : Point) : HitTestInfo;
22
```

#### Description

20 Gets information, such as row and column number of a clicked point on the grid,  
21 about the grid using a specific **System.Drawing.Point**.

22 *Return Value:* A **System.Windows.Forms.DataGrid.HitTestInfo** that  
23 contains specific information about the grid.

24 The **System.Windows.Forms.DataGrid.HitTestInfo**, in conjunction with the  
25 **System.Windows.Forms.DataGrid.HitTest(System.Int32, System.Int32)**  
method of the **System.Windows.Forms.DataGrid** control, is used to  
determine which part of a **System.Windows.Forms.DataGrid** control the user



has clicked. The **System.Windows.Forms.DataGrid.HitTestInfo** contains both the row, column and part of the grid that was clicked. Additionally, the **System.Windows.Forms.DataGrid.HitTestInfo.Type** property returns a **System.Windows.Forms.DataGrid.HitTestType** enumeration. A **System.Drawing.Point** that represents single x,y coordinate.

*vvvvvvvv)HitTest*

[C#]        public        HitTestInfo        HitTest(int        x,        int        y);

[C++]        public:        HitTestInfo\*        HitTest(int        x,        int        y);

[VB] Public Function HitTest(ByVal x As Integer, ByVal y As Integer) As HitTestInfo

[JScript] public function HitTest(x : int, y : int) : HitTestInfo; Gets information about the **System.Windows.Forms.DataGrid** control at a specified point on the screen.

#### *Description*

Gets information, such as row and column number of a clicked point on the grid, using the x and y coordinate passed to the method.

*Return Value:* A **System.Windows.Forms.DataGrid.HitTestInfo** that contains information about the clicked part of the grid.

The **System.Windows.Forms.DataGrid.HitTestInfo** , in conjunction with the **System.Windows.Forms.DataGrid.HitTest(System.Int32, System.Int32)** method of the **System.Windows.Forms.DataGrid** control, is used to determine which part of a **System.Windows.Forms.DataGrid** control the user has clicked. The **System.Windows.Forms.DataGrid.HitTestInfo** contains both the row, column and part of the grid that was clicked. Additionally, the **System.Windows.Forms.DataGrid.HitTestInfo.Type** property returns a **System.Windows.Forms.DataGrid.HitTestType** enumeration. The horizontal position of the coordinate. The vertical position of the coordinate.

## wwwwww)IsExpanded

```
1
2 [C#]      public      bool      IsExpanded(int      rowNumber);
3
4 [C++]      public:      bool      IsExpanded(int      rowNumber);
5
6 [VB] Public Function IsExpanded(ByVal rowNumber As Integer) As Boolean
7
8 [JScript] public  function  IsExpanded(rowNumber : int) : Boolean;
```

### Description

Gets a value that indicates whether a specified row's node is expanded or collapsed.

*Return Value:* **true** if the node is expanded; otherwise, **false** . The number of the row in question.

## xxxxxxx)IsSelected

```
13 [C#]      public      bool      IsSelected(int      row);
14
15 [C++]      public:      bool      IsSelected(int      row);
16
17 [VB] Public Function IsSelected(ByVal row As Integer) As Boolean
18
19 [JScript] public  function  IsSelected(row : int) : Boolean;
```

### Description

Gets a value indicating whether a specified row is selected.

*Return Value:* **true** if the row is selected; otherwise, **false** . The number of the row you interested in.

## yyyyyyyy)NavigateBack

```
23 [C#]      public      void      NavigateBack();
24
25 [C++]      public:      void      NavigateBack();
```

|           |        |          |                 |
|-----------|--------|----------|-----------------|
| [VB]      | Public | Sub      | NavigateBack()  |
| [JScript] | public | function | NavigateBack(); |

#### *Description*

Navigates back to the table previously displayed in the grid.

If the grid has no parent rows, no change occurs.

#### *zzzzzzzz)NavigateTo*

[C#] public void NavigateTo(int rowNum, string relationName);

[C++] public: void NavigateTo(int rowNum, String\* relationName);

[VB] Public Sub NavigateTo(ByVal rowNum As Integer, ByVal relationName As String)

[JScript] public function NavigateTo(rowNum : int, relationName : String);

Navigates to a specific table that displays in the  
**System.Windows.Forms.DataGrid** control.

#### *Description*

Navigates to the table specified by row and relation name. The number of the row to navigate to. The name of the child relation to navigate to.

#### *aaaaaaaa)OnAllowNavigationChanged*

[C#] protected virtual void OnAllowNavigationChanged(EventArgs e);

[C++] protected: virtual void OnAllowNavigationChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnAllowNavigationChanged(ByVal e As EventArgs)

1 [JScript] protected function OnAllowNavigationChanged(e : EventArgs);

3 *Description*

4 Raises the **System.Windows.Forms.DataGrid.AllowNavigationChanged**  
event. An **System.EventArgs** that contains the event data.

5 *bbbbbbbbb)OnBackButtonClicked*

7 [C#] protected void OnBackButtonClicked(object sender, EventArgs e);

8 [C++] protected: void OnBackButtonClicked(Object\* sender, EventArgs\* e);

9 [VB] Protected Sub OnBackButtonClicked(ByVal sender As Object, ByVal e As  
10 EventArgs)

11 [JScript] protected function OnBackButtonClicked(sender : Object, e :  
12 EventArgs);

14 *Description*

15 Listens for the caption's back button clicked event.

16 Raising an event invokes the event handler through a delegate. For more  
17 information, see . An **System.Object** that contains data about the control. An  
**System.EventArgs** that contains data about the event.

18 *ccccccccc)OnBackColorChanged*

20 [C#] protected override void OnBackColorChanged(EventArgs e);

21 [C++] protected: void OnBackColorChanged(EventArgs\* e);

22 [VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)

23 [JScript] protected override function OnBackColorChanged(e : EventArgs);

1  
2 *Description*

3 Raises the **System.Windows.Forms.Control.BackColorChanged** event.

4 Raising an event invokes the event handler through a delegate. For more  
5 information, see . An **System.EventArgs** that contains the event data.

6 *dddddddd)OnBackgroundColorChanged*

7 [C#] protected virtual void OnBackgroundColorChanged(EventArgs e);

8 [C++] protected: virtual void OnBackgroundColorChanged(EventArgs\* e);

9 [VB] Overridable Protected Sub OnBackgroundColorChanged(ByVal e As  
10 EventArgs)

11 [JScript] protected function OnBackgroundColorChanged(e : EventArgs);

12  
13 *Description*

14 Raises the **System.Windows.Forms.DataGrid.BackgroundColorChanged**  
15 event. An **System.EventArgs** that contains the event data.

16 *eeeeeeee)OnBindingContextChanged*

17  
18 [C#] protected override void OnBindingContextChanged(EventArgs e);

19 [C++] protected: void OnBindingContextChanged(EventArgs\* e);

20 [VB] Overrides Protected Sub OnBindingContextChanged(ByVal e As  
21 EventArgs)

22 [JScript] protected override function OnBindingContextChanged(e : EventArgs);

23  
24 *Description*

1 Raises the **System.Windows.Forms.Control.BindingContextChanged**  
event.

2 Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

3  
4 *ffffffff)OnBorderStyleChanged*

5  
6 [C#] protected virtual void OnBorderStyleChanged(EventArgs e);

7 [C++] protected: virtual void OnBorderStyleChanged(EventArgs\* e);

8 [VB] Overridable Protected Sub OnBorderStyleChanged(ByVal e As EventArgs)

9 [JScript] protected function OnBorderStyleChanged(e : EventArgs);

10  
11 *Description*

12 Raises the **System.Windows.Forms.DataGrid.BorderStyleChanged** event.  
An **System.EventArgs** that contains the event data.

13  
14 *ggggggggg)OnCaptionVisibleChanged*

15 [C#] protected virtual void OnCaptionVisibleChanged(EventArgs e);

16 [C++] protected: virtual void OnCaptionVisibleChanged(EventArgs\* e);

17 [VB] Overridable Protected Sub OnCaptionVisibleChanged(ByVal e As  
18 EventArgs)

19 [JScript] protected function OnCaptionVisibleChanged(e : EventArgs);

20  
21 *Description*

22 Raises the **System.Windows.Forms.DataGrid.CaptionVisibleChanged**  
23 event. An **System.EventArgs** that contains the event data.

### *hhhhhhhh)OnCurrentCellChanged*

```
[C#]    protected    virtual    void    OnCurrentCellChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnCurrentCellChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnCurrentCellChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnCurrentCellChanged(e    :    EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.DataGrid.CurrentCellChanged** event. An **System.EventArgs** that contains the event data.

### *iiiiiii) OnDataSourceChanged*

```
[C#]    protected    virtual    void    OnDataSourceChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnDataSourceChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnDataSourceChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnDataSourceChanged(e    :    EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.DataGrid.DataSourceChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *jjjjjjj) OnEnter*

```
[C#]    protected    override    void    OnEnter(EventArgs    e);
[C++]    protected:    void    OnEnter(EventArgs*    e);
[VB]    Overrides Protected Sub OnEnter(ByVal e As EventArgs)
```

[JScript] protected override function OnEnter(e : EventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.Enter** event. An **System.EventArgs** that contains the event data.

### *kkkkkkkkk)OnFlatModeChanged*

[C#] protected virtual void OnFlatModeChanged(EventArgs e);

[C++] protected: virtual void OnFlatModeChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnFlatModeChanged(ByVal e As EventArgs)

[JScript] protected function OnFlatModeChanged(e : EventArgs);

### *Description*

Raises the **System.Windows.Forms.DataGrid.FlatModeChanged** event. An **System.EventArgs** that contains the event data.

### *lllllllll) OnFontChanged*

[C#] protected override void OnFontChanged(EventArgs e);

[C++] protected: void OnFontChanged(EventArgs\* e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.FontChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.



### *mmmmmmmmmm)OnForeColorChanged*

[C#] protected override void OnForeColorChanged(EventArgs e);  
[C++] protected: void OnForeColorChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnForeColorChanged(ByVal e As EventArgs)  
[JScript] protected override function OnForeColorChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.ForeColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *nnnnnnnnnn)OnHandleCreated*

[C#] protected override void OnHandleCreated(EventArgs e);  
[C++] protected: void OnHandleCreated(EventArgs\* e);  
[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)  
[JScript] protected override function OnHandleCreated(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.CreateHandle** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *oooooooooo)OnHandleDestroyed*

[C#] protected override void OnHandleDestroyed(EventArgs e);  
[C++] protected: void OnHandleDestroyed(EventArgs\* e);

[VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

[JScript] protected override function OnHandleDestroyed(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.DestroyHandle** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** containing the event data.

#### *OnKeyDown*

[C#] protected override void OnKeyDown(KeyEventArgs ke);

[C++] protected: void OnKeyDown(KeyEventArgs\* ke);

[VB] Overrides Protected Sub OnKeyDown(ByVal ke As KeyEventArgs)

[JScript] protected override function OnKeyDown(ke : KeyEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.KeyDown** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyEventArgs** that provides data about the

**System.Windows.Forms.Control.OnKeyDown(System.Windows.Forms.KeyEventArgs)** event.

#### *OnKeyPress*

[C#] protected override void OnKeyPress(KeyPressEventArgs kpe);

[C++] protected: void OnKeyPress(KeyPressEventArgs\* kpe);

[VB] Overrides Protected Sub OnKeyPress(ByVal kpe As KeyPressEventArgs)

[JScript] protected override function OnKeyPress(kpe : KeyPressEventArgs);

## Description

Raises the **System.Windows.Forms.Control.KeyPress** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyPressEventArgs** that contains data about the

**System.Windows.Forms.Control.OnKeyPress(System.Windows.Forms.KeyPressEventArgs)** event

## *rrrrrrrr)OnLayout*

[C#] protected override void OnLayout(LayoutEventArgs levent);

[C++] protected: void OnLayout(LayoutEventArgs\* levent);

[VB] Overrides Protected Sub OnLayout(ByVal levent As LayoutEventArgs)

[JScript] protected override function OnLayout(levent : LayoutEventArgs);

## Description

Raises the **System.Windows.Forms.Control.Layout** event that repositions controls and updates scroll bars.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.LayoutEventArgs** that contains the event data.

## *ssssssss)OnLeave*

[C#] protected override void OnLeave(EventArgs e);

[C++] protected: void OnLeave(EventArgs\* e);

[VB] Overrides Protected Sub OnLeave(ByVal e As EventArgs)

[JScript] protected override function OnLeave(e : EventArgs);

## Description

Raises the **System.Windows.Forms.Control.Leave** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *tttttttt) OnMouseDown*

[C#]   protected   override   void   OnMouseDown(MouseEventArgs   e);

[C++]   protected:   void   OnMouseDown(MouseEventArgs\*   e);

[VB] Overrides Protected Sub OnMouseDown(ByVal e As MouseEventArgs)

[JScript] protected override function OnMouseDown(e : MouseEventArgs);

## Description

Raises the **System.Windows.Forms.Control.MouseDown** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.MouseEventArgs** that contains data about the

**System.Windows.Forms.Control.OnMouseDown(System.Windows.Forms.MouseEventArgs)** event.

### *uuuuuuuuuu)OnMouseLeave*

[C#]   protected   override   void   OnMouseLeave(EventArgs   e);

[C++]   protected:   void   OnMouseLeave(EventArgs\*   e);

[VB] Overrides Protected Sub OnMouseLeave(ByVal e As EventArgs)

[JScript] protected override function OnMouseLeave(e : EventArgs);

## Description

Creates the **System.Windows.Forms.Control.MouseLeave** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains data about the **System.Windows.Forms.Control.OnMouseLeave(System.EventArgs)** event.

*vvvvvvvvv)OnMouseMove*

[C#] protected override void OnMouseMove(MouseEventArgs e);

[C++] protected: void OnMouseMove(MouseEventArgs\* e);

[VB] Overrides Protected Sub OnMouseMove(ByVal e As MouseEventArgs)

[JScript] protected override function OnMouseMove(e : MouseEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.MouseMove** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.MouseEventArgs** that contains data about the **System.Windows.Forms.Control.OnMouseMove(System.Windows.Forms.MouseEventArgs)** event.

*wwwwwwwww)OnMouseUp*

[C#] protected override void OnMouseUp(MouseEventArgs e);

[C++] protected: void OnMouseUp(MouseEventArgs\* e);

[VB] Overrides Protected Sub OnMouseUp(ByVal e As MouseEventArgs)

[JScript] protected override function OnMouseUp(e : MouseEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.MouseUp** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.MouseEventArgs** that contains data about the

**System.Windows.Forms.Control.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event.

*xxxxxxxxx)OnMouseWheel*

[C#]   protected   override   void   OnMouseWheel(MouseEventArgs   e);

[C++]   protected:   void   OnMouseWheel(MouseEventArgs\*   e);

[VB] Overrides Protected Sub OnMouseWheel(ByVal e As MouseEventArgs)

[JScript] protected override function OnMouseWheel(e : MouseEventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.MouseWheel** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.MouseEventArgs** that contains data about the

**System.Windows.Forms.Control.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event.

*yyyyyyyyy)OnNavigate*

[C#]   protected   void   OnNavigate(NavigateEventArgs   e);

[C++]   protected:   void   OnNavigate(NavigateEventArgs\*   e);

[VB] Protected Sub OnNavigate(ByVal e As NavigateEventArgs)

[JScript] protected function OnNavigate(e : NavigateEventArgs);

#### *Description*

Raises the **System.Windows.Forms.DataGrid.Navigate** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.NavigateEventArgs** that contains the event data.

### *zzzzzzzzzz)OnPaint*

[C#]      protected      override      void      OnPaint(PaintEventArgs      pe);

[C++]      protected:      void      OnPaint(PaintEventArgs\*      pe);

[VB]      Overrides      Protected      Sub      OnPaint(ByVal pe As PaintEventArgs)

[JScript]      protected      override      function      OnPaint(pe : PaintEventArgs);

### *Description*

Raises the **System.Windows.Forms.Control.Paint** event.

Raising an event invokes the event handler through a delegate. For an overview, see . A **System.Windows.Forms.PaintEventArgs** which contains data about the event.

### *aaaaaaaaaaaa)OnPaintBackground*

[C#]      protected      override      void      OnPaintBackground(PaintEventArgs      ebe);

[C++]      protected:      void      OnPaintBackground(PaintEventArgs\*      ebe);

[VB]      Overrides      Protected      Sub      OnPaintBackground(ByVal ebe As PaintEventArgs)

[JScript]      protected      override      function      OnPaintBackground(ebe : PaintEventArgs);

### *Description*

Overrides **System.Windows.Forms.Control.OnPaintBackground(System.Windows.Forms.PaintEventArgs)** to prevent painting the background of the **System.Windows.Forms.DataGrid** control.

Because the **System.Windows.Forms.DataGrid** is a complex control, this override is implemented to have no action. Therefore, calling this method will have no effect. A **System.Windows.Forms.PaintEventArgs** that contains information about the control to paint.

#### *bbbbbbbbbb)OnParentRowsLabelStyleChanged*

[C#] protected virtual void OnParentRowsLabelStyleChanged(EventArgs e);

[C++] protected: virtual void OnParentRowsLabelStyleChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentRowsLabelStyleChanged(ByVal e As EventArgs)

[JScript] protected function OnParentRowsLabelStyleChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DataGrid.ParentRowsLabelStyleChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *ccccccccc)OnParentRowsVisibleChanged*

[C#] protected virtual void OnParentRowsVisibleChanged(EventArgs e);

[C++] protected: virtual void OnParentRowsVisibleChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnParentRowsVisibleChanged(ByVal e As EventArgs)

[JScript] protected function OnParentRowsVisibleChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DataGrid.ParentRowsVisibleChanged** event. An **System.EventArgs** that contains the event data.



## *dddddddddd)OnReadOnlyChanged*

```
[C#]    protected    virtual    void    OnReadOnlyChanged(EventArgs    e);
[C++]    protected:    virtual    void    OnReadOnlyChanged(EventArgs*    e);
[VB]    Overridable Protected Sub OnReadOnlyChanged(ByVal e As EventArgs)
[JScript]    protected    function    OnReadOnlyChanged(e    :    EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.DataGrid.ReadOnlyChanged** event

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## *eeeeeeeeee)OnResize*

```
[C#]    protected    override    void    OnResize(EventArgs    e);
[C++]    protected:    void    OnResize(EventArgs*    e);
[VB]    Overrides Protected Sub OnResize(ByVal e As EventArgs)
[JScript]    protected    override    function    OnResize(e    :    EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Control.Resize** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains the event data.

## *ffffffff)OnRowHeaderClick*

```
[C#]    protected    void    OnRowHeaderClick(EventArgs    e);
[C++]    protected:    void    OnRowHeaderClick(EventArgs*    e);
```

1 [VB] Protected Sub OnRowHeaderClick(ByVal e As EventArgs)

2 [JScript] protected function OnRowHeaderClick(e : EventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.DataGrid.RowHeaderClick** event.

6 Raising an event invokes the event handler through a delegate. For more  
7 information, see . An **System.EventArgs** that contains the event data.

8 *gggggggggg)OnScroll*

9 [C#] protected void OnScroll(EventArgs e);

10 [C++] protected: void OnScroll(EventArgs\* e);

11 [VB] Protected Sub OnScroll(ByVal e As EventArgs)

12 [JScript] protected function OnScroll(e : EventArgs);

13  
14 *Description*

15 Raises the **System.Windows.Forms.DataGrid.Scroll** event.

16 Raising an event invokes the event handler through a delegate. For more  
17 information, see . An **System.EventArgs** that contains the event data.

18 *hhhhhhhhh)OnShowParentDetailsButtonClicked*

19  
20 [C#] protected void OnShowParentDetailsButtonClicked(object sender, EventArgs  
21 e);

22 [C++] protected: void OnShowParentDetailsButtonClicked(Object\* sender,  
23 EventArgs\* e);

24 [VB] Protected Sub OnShowParentDetailsButtonClicked(ByVal sender As Object,  
25 ByVal e As EventArgs)

[JScript] protected function OnShowParentDetailsButtonClicked(sender : Object,  
e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DataGrid.ShowParentDetailsButtonClick** event.  
The source of the event. An **System.EventArgs** that contains the event data.

#### *iiiiiii)ProcessDialogKey*

[C#] protected override bool ProcessDialogKey(Keys keyData);

[C++] protected: bool ProcessDialogKey(Keys keyData);

[VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)

As Boolean

[JScript] protected override function ProcessDialogKey(keyData : Keys) :

Boolean;

#### *Description*

Gets or sets a value that indicates whether a key should be processed further.  
*Return Value:* **true** , the key should be processed; otherwise, **false** .

The method overrides the **System.Windows.Forms.Control.ProcessDialogKey(System.Windows.Forms.Keys)** method to implement keyboard navigation of the grid. A **System.Windows.Forms.Keys** that contains data about the pressed key.

#### *jjjjjjj)ProcessGridKey*

[C#] protected bool ProcessGridKey(KeyEventArgs ke);

[C++] protected: bool ProcessGridKey(KeyEventArgs\* ke);

1 [VB] Protected Function ProcessGridKey(ByVal ke As KeyEventArgs) As  
2 Boolean

3 [JScript] protected function ProcessGridKey(ke : KeyEventArgs) : Boolean;

4  
5 *Description*

6 Processes keys for grid navigation.

*Return Value:* **true** , if the key was processed; otherwise **false** . A

7 **System.Windows.Forms.KeyEventArgs** that contains data about the key up  
8 or key down event.

9 *kkkkkkkkkk)ProcessKeyPreview*

10 [C#] protected override bool ProcessKeyPreview(ref Message m);

11 [C++] protected: bool ProcessKeyPreview(Message\* m);

12 [VB] Overrides Protected Function ProcessKeyPreview(ByRef m As Message) As  
13 Boolean

14 [JScript] protected override function ProcessKeyPreview(m : Message) : Boolean;

15  
16 *Description*

17 Previews a keyboard message and returns a value indicating if the key was  
18 consumed.

*Return Value:* **true** , if the key was consumed; otherwise, **false** .

19 This method is called by a child control when the child control receives a  
20 keyboard message. The child control calls this method before generating any  
21 keyboard events for the message. If this method returns **true** , the child control  
22 considers the message consumed and does not generate any keyboard events. A  
23 **System.Windows.Forms.Message** that contains data about the event. The  
24 parameter is passed by reference.  
25

### |||||||)ProcessTabKey

```
[C#]      protected      bool      ProcessTabKey(Keys      keyData);
[C++]     protected:     bool      ProcessTabKey(Keys      keyData);
[VB] Protected Function ProcessTabKey(ByVal keyData As Keys) As Boolean
[JScript] protected function ProcessTabKey(keyData : Keys) : Boolean;
```

#### Description

Gets a value indicating whether the Tab key should be processed.

**Return Value:** **true** if the Tab key should be processed; otherwise, **false** . A **System.Windows.Forms.Keys** that contains data about which the pressed key.

### aaaaaaaaaaaa)ResetAlternatingBackColor

```
[C#]      public          void          ResetAlternatingBackColor();
[C++]     public:         void          ResetAlternatingBackColor();
[VB]      Public          Sub          ResetAlternatingBackColor()
[JScript] public          function      ResetAlternatingBackColor();
```

#### Description

Resets the **System.Windows.Forms.DataGrid.AlternatingBackColor** property to its default color.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeAlternatingBackColor** method to determine whether the property value has changed from its default.

*nnnnnnnnnn)ResetBackColor*

|           |           |          |          |                   |
|-----------|-----------|----------|----------|-------------------|
| [C#]      | public    | override | void     | ResetBackColor(); |
| [C++]     | public:   |          | void     | ResetBackColor(); |
| [VB]      | Overrides | Public   | Sub      | ResetBackColor()  |
| [JScript] | public    | override | function | ResetBackColor(); |

*Description*

Resets the **System.Windows.Forms.DataGrid.BackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*oooooooooooo)ResetForeColor*

|           |           |          |          |                   |
|-----------|-----------|----------|----------|-------------------|
| [C#]      | public    | override | void     | ResetForeColor(); |
| [C++]     | public:   |          | void     | ResetForeColor(); |
| [VB]      | Overrides | Public   | Sub      | ResetForeColor()  |
| [JScript] | public    | override | function | ResetForeColor(); |

*Description*

Resets the **System.Windows.Forms.DataGrid.ForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

pppppppppp)ResetGridLineColor

|           |         |          |                       |
|-----------|---------|----------|-----------------------|
| [C#]      | public  | void     | ResetGridLineColor(); |
| [C++]     | public: | void     | ResetGridLineColor(); |
| [VB]      | Public  | Sub      | ResetGridLineColor()  |
| [JScript] | public  | function | ResetGridLineColor(); |

Description

Resets the **System.Windows.Forms.DataGrid.GridLineColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeGridLineColor** method to determine whether the property value has changed from its default.

qqqqqqqqqq)ResetHeaderBackColor

|           |         |          |                         |
|-----------|---------|----------|-------------------------|
| [C#]      | public  | void     | ResetHeaderBackColor(); |
| [C++]     | public: | void     | ResetHeaderBackColor(); |
| [VB]      | Public  | Sub      | ResetHeaderBackColor()  |
| [JScript] | public  | function | ResetHeaderBackColor(); |

Description

Resets the **System.Windows.Forms.DataGrid.HeaderBackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeHeaderBackColor** method to determine whether the property value has changed from its default.

### *rrrrrrrrr)ResetHeaderFont*

|           |         |          |                    |
|-----------|---------|----------|--------------------|
| [C#]      | public  | void     | ResetHeaderFont(); |
| [C++]     | public: | void     | ResetHeaderFont(); |
| [VB]      | Public  | Sub      | ResetHeaderFont()  |
| [JScript] | public  | function | ResetHeaderFont(); |

#### *Description*

Resets the **System.Windows.Forms.DataGrid.HeaderFont** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeHeaderFont** method to determine whether the property value has changed from its default.

### *sssssssss)ResetHeaderForeColor*

|           |         |          |                         |
|-----------|---------|----------|-------------------------|
| [C#]      | public  | void     | ResetHeaderForeColor(); |
| [C++]     | public: | void     | ResetHeaderForeColor(); |
| [VB]      | Public  | Sub      | ResetHeaderForeColor()  |
| [JScript] | public  | function | ResetHeaderForeColor(); |

#### *Description*

Resets the **System.Windows.Forms.DataGrid.HeaderForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeHeaderForeColor** method to determine whether the property value has changed from its default.



### *tttttttt)ResetLinkColor*

|           |         |          |                   |
|-----------|---------|----------|-------------------|
| [C#]      | public  | void     | ResetLinkColor(); |
| [C++]     | public: | void     | ResetLinkColor(); |
| [VB]      | Public  | Sub      | ResetLinkColor()  |
| [JScript] | public  | function | ResetLinkColor(); |

#### *Description*

Resets the **System.Windows.Forms.DataGrid.LinkColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

### *uuuuuuuuuu)ResetLinkHoverColor*

|           |         |          |                        |
|-----------|---------|----------|------------------------|
| [C#]      | public  | void     | ResetLinkHoverColor(); |
| [C++]     | public: | void     | ResetLinkHoverColor(); |
| [VB]      | Public  | Sub      | ResetLinkHoverColor()  |
| [JScript] | public  | function | ResetLinkHoverColor(); |

#### *Description*

Resets the **System.Windows.Forms.DataGrid.LinkHoverColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeLinkHoverColor** method to determine whether the property value has changed from its default.

vvvvvvvvvv)ResetSelection

|           |            |          |                   |
|-----------|------------|----------|-------------------|
| [C#]      | protected  | void     | ResetSelection(); |
| [C++]     | protected: | void     | ResetSelection(); |
| [VB]      | Protected  | Sub      | ResetSelection()  |
| [JScript] | protected  | function | ResetSelection(); |

*Description*

Turns off selection for all rows that are selected.

wwwwwwwwww)ResetSelectionBackColor

|           |         |          |                            |
|-----------|---------|----------|----------------------------|
| [C#]      | public  | void     | ResetSelectionBackColor(); |
| [C++]     | public: | void     | ResetSelectionBackColor(); |
| [VB]      | Public  | Sub      | ResetSelectionBackColor()  |
| [JScript] | public  | function | ResetSelectionBackColor(); |

*Description*

Resets the **System.Windows.Forms.DataGrid.SelectionBackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeSelectionBackColor** method to determine whether the property value has changed from its default.

xxxxxxxxxx)ResetSelectionForeColor

|      |        |      |                            |
|------|--------|------|----------------------------|
| [C#] | public | void | ResetSelectionForeColor(); |
|------|--------|------|----------------------------|

|           |         |          |                            |
|-----------|---------|----------|----------------------------|
| [C++]     | public: | void     | ResetSelectionForeColor(); |
| [VB]      | Public  | Sub      | ResetSelectionForeColor()  |
| [JScript] | public  | function | ResetSelectionForeColor(); |

#### Description

Resets the **System.Windows.Forms.DataGrid.SelectionForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** . You can use the **System.Windows.Forms.DataGrid.ShouldSerializeSelectionForeColor** method to determine whether the property value has changed from its default.

yyyyyyyyyy)Select

|           |         |          |              |                 |
|-----------|---------|----------|--------------|-----------------|
| [C#]      | public  | void     | Select(int   | row);           |
| [C++]     | public: | void     | Select(int   | row);           |
| [VB]      | Public  | Sub      | Select(ByVal | row As Integer) |
| [JScript] | public  | function | Select(row   | : int);         |

#### Description

Selects a specified row. The index of the row to select.

zzzzzzzzzz)SetDataBinding

|       |            |      |                        |             |         |               |
|-------|------------|------|------------------------|-------------|---------|---------------|
| [C#]  | public     | void | SetDataBinding(object  | dataSource, | string  | dataMember);  |
| [C++] | public:    | void | SetDataBinding(Object* | dataSource, | String* | dataMember);  |
| [VB]  | Public     | Sub  | SetDataBinding(ByVal   | dataSource  | As      | Object, ByVal |
|       | dataMember |      | As                     |             |         | String)       |

1 [JScript] public function SetDataBinding(dataSource : Object, dataMember :  
2 String);

#### 3 4 *Description*

5 Sets the **System.Windows.Forms.DataGrid.DataSource** and  
6 **System.Windows.Forms.DataGrid.DataMember** properties at run time.

7 You must use the  
8 **System.Windows.Forms.DataGrid.SetDataBinding(System.Object, System.  
9 m.String)** method at run time to reset the  
10 **System.Windows.Forms.DataGrid.DataSource** property. The data source,  
11 typed as **System.Object**, for the **System.Windows.Forms.DataGrid** control.  
12 The **System.Windows.Forms.DataGrid.DataMember** string that specifies  
13 the table to bind to within the object returned by the  
14 **System.Windows.Forms.DataGrid.DataSource** property.

15  
16 *aaaaaaaaaaaa)ShouldSerializeAlternatingBackColor*

17 [C#] protected virtual bool ShouldSerializeAlternatingBackColor();

18 [C++] protected: virtual bool ShouldSerializeAlternatingBackColor();

19 [VB] Overridable Protected Function ShouldSerializeAlternatingBackColor() As  
20 Boolean

21 [JScript] protected function ShouldSerializeAlternatingBackColor() : Boolean;

#### 22 23 *Description*

24 Indicates whether the  
25 **System.Windows.Forms.DataGrid.AlternatingBackColor** property should  
be persisted.  
*Return Value:* **true** if the property value has changed from its default; otherwise,  
**false** .

You typically use this method if you are either creating a designer for the  
**System.Windows.Forms.DataGrid** or creating your own control incorporating  
the **System.Windows.Forms.DataGrid** .

### *bbbbbbbbbbb)ShouldSerializeBackgroundColor*

[C#]      protected      virtual      bool      ShouldSerializeBackgroundColor();  
[C++]      protected:      virtual      bool      ShouldSerializeBackgroundColor();  
[VB]      Overridable      Protected      Function      ShouldSerializeBackgroundColor()      As  
Boolean  
[JScript]      protected      function      ShouldSerializeBackgroundColor()      :      Boolean;

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.BackgroundColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

### *ccccccccccc)ShouldSerializeCaptionBackColor*

[C#]      protected      virtual      bool      ShouldSerializeCaptionBackColor();  
[C++]      protected:      virtual      bool      ShouldSerializeCaptionBackColor();  
[VB]      Overridable      Protected      Function      ShouldSerializeCaptionBackColor()      As  
Boolean  
[JScript]      protected      function      ShouldSerializeCaptionBackColor()      :      Boolean;

#### *Description*

Gets a value indicating whether the **System.Windows.Forms.DataGrid.CaptionBackColor** property should be persisted.

*Return Value:* **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*ddddddddddd)ShouldSerializeCaptionForeColor*

[C#]      protected      virtual      bool      ShouldSerializeCaptionForeColor();

[C++]      protected:      virtual      bool      ShouldSerializeCaptionForeColor();

[VB]      Overridable      Protected      Function      ShouldSerializeCaptionForeColor()      As      Boolean

[JScript]      protected      function      ShouldSerializeCaptionForeColor()      :      Boolean;

#### *Description*

Gets a value indicating whether the **System.Windows.Forms.DataGrid.CaptionForeColor** property should be persisted.

*Return Value:* **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*eeeeeeeeeee)ShouldSerializeGridLineColor*

[C#]      protected      virtual      bool      ShouldSerializeGridLineColor();

[C++]      protected:      virtual      bool      ShouldSerializeGridLineColor();

[VB]      Overridable      Protected      Function      ShouldSerializeGridLineColor()      As      Boolean

[JScript]      protected      function      ShouldSerializeGridLineColor()      :      Boolean;

*Description*

Indicates whether the **System.Windows.Forms.DataGrid.GridLineColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*ShouldSerializeHeaderBackColor*

[C#]       protected       virtual       bool       ShouldSerializeHeaderBackColor();

[C++]       protected:       virtual       bool       ShouldSerializeHeaderBackColor();

[VB]   Overridable   Protected   Function   ShouldSerializeHeaderBackColor()   As  
Boolean

[JScript]   protected   function   ShouldSerializeHeaderBackColor()   :   Boolean;

*Description*

Indicates whether the **System.Windows.Forms.DataGrid.HeaderBackColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*ShouldSerializeHeaderFont*

[C#]       protected       bool       ShouldSerializeHeaderFont();

[C++]       protected:       bool       ShouldSerializeHeaderFont();

```

1 [VB] Protected Function ShouldSerializeHeaderFont() As Boolean
2 [JScript] protected function ShouldSerializeHeaderFont() : Boolean;

```

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.HeaderFont** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*hhhhhhhhhh)ShouldSerializeHeaderForeColor*

```

11 [C#] protected virtual bool ShouldSerializeHeaderForeColor();
12 [C++] protected: virtual bool ShouldSerializeHeaderForeColor();
13 [VB] Overridable Protected Function ShouldSerializeHeaderForeColor() As
14 Boolean
15 [JScript] protected function ShouldSerializeHeaderForeColor() : Boolean;

```

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.HeaderForeColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .



#### *iiiiiiiiii)ShouldSerializeLinkHoverColor*

```
1
2 [C#]    protected    virtual    bool    ShouldSerializeLinkHoverColor();
3
4 [C++]    protected:    virtual    bool    ShouldSerializeLinkHoverColor();
5
6 [VB]    Overridable    Protected    Function    ShouldSerializeLinkHoverColor() As
7 Boolean
8
9 [JScript]    protected    function    ShouldSerializeLinkHoverColor() : Boolean;
```

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.LinkHoverColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

#### *jjjjjjjjjj)ShouldSerializeParentRowsBackColor*

```
14
15
16 [C#]    protected    virtual    bool    ShouldSerializeParentRowsBackColor();
17
18 [C++]    protected:    virtual    bool    ShouldSerializeParentRowsBackColor();
19
20 [VB]    Overridable    Protected    Function    ShouldSerializeParentRowsBackColor() As
21 Boolean
22
23 [JScript]    protected    function    ShouldSerializeParentRowsBackColor() : Boolean;
```

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.ParentRowsBackColor** property should be persisted.

*Return Value:* **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*kkkkkkkkkk)ShouldSerializeParentRowsForeColor*

[C#]      protected      virtual      bool      ShouldSerializeParentRowsForeColor();

[C++]      protected:      virtual      bool      ShouldSerializeParentRowsForeColor();

[VB] Overridable Protected Function ShouldSerializeParentRowsForeColor() As Boolean

[JScript] protected function ShouldSerializeParentRowsForeColor() : Boolean;

#### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.ParentRowsForeColor** property should be persisted.

*Return Value:* **true** if the property value has been changed from its default; otherwise, **false** .

You typically use this method only if you are either creating a designer for the **System.Windows.Forms.DataGrid** , or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*lllllllll)ShouldSerializePreferredRowHeight*

[C#]              protected              bool              ShouldSerializePreferredRowHeight();

[C++]              protected:              bool              ShouldSerializePreferredRowHeight();

[VB] Protected Function ShouldSerializePreferredRowHeight() As Boolean

[JScript] protected function ShouldSerializePreferredRowHeight() : Boolean;

*Description*

Indicates whether the **System.Windows.Forms.DataGrid.PreferredRowHeight** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*mmmmmmmmmmmm)ShouldSerializeSelectionBackColor*

[C#]           protected           bool           ShouldSerializeSelectionBackColor();

[C++]           protected:           bool           ShouldSerializeSelectionBackColor();

[VB]   Protected   Function   ShouldSerializeSelectionBackColor()   As   Boolean

[JScript]   protected   function   ShouldSerializeSelectionBackColor() : Boolean;

*Description*

Indicates whether the **System.Windows.Forms.DataGrid.SelectionBackColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid** .

*nnnnnnnnnnnn)ShouldSerializeSelectionForeColor*

[C#]           protected           virtual       bool           ShouldSerializeSelectionForeColor();

[C++]           protected:           virtual       bool           ShouldSerializeSelectionForeColor();

[VB] Overridable Protected Function ShouldSerializeSelectionForeColor() As Boolean

[JScript] protected function ShouldSerializeSelectionForeColor() : Boolean;

### *Description*

Indicates whether the **System.Windows.Forms.DataGrid.SelectionForeColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false**.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid** or creating your own control incorporating the **System.Windows.Forms.DataGrid**.

*oooooooooooo)SubObjectsSiteChange*

[C#] public void SubObjectsSiteChange(bool site);

[C++] public: void SubObjectsSiteChange(bool site);

[VB] Public Sub SubObjectsSiteChange(ByVal site As Boolean)

[JScript] public function SubObjectsSiteChange(site : Boolean);

*pppppppppppp)UnSelect*

[C#] public void UnSelect(int row);

[C++] public: void UnSelect(int row);

[VB] Public Sub UnSelect(ByVal row As Integer)

[JScript] public function UnSelect(row : int);

### *Description*

Unselects a specified row. The index of the row to deselect.

DataGridBoolColumn class (System.Windows.Forms)

a) *WndProc*

### *Description*

Specifies a column in which each cell contains a check box for representing a Boolean value.

The **System.Windows.Forms.DataGridBoolColumn** derives from the **abstract** class **System.Windows.Forms.DataGridColumnStyle** . At run time, the **System.Windows.Forms.DataGridBoolColumn** contains check boxes in each cell that have three states: checked ( **true** ), unchecked ( **false** ), and **System.DBNull.Value** .

b) *DataGridBoolColumn*

*Example Syntax:*

c) *WndProc*

[C#]                      public                      DataGridBoolColumn();

[C++]                      public:                      DataGridBoolColumn();

[VB]                      Public                      Sub                      New()

[JScript] public function DataGridBoolColumn(); Initializes a new instance of the

**System.Windows.Forms.DataGridBoolColumn** class.

### *Description*

Initializes a new instance of the **System.Windows.Forms.DataGridBoolColumn** class.

When using this overload to create a **System.Windows.Forms.DataGridBoolColumn** , be sure to set the **System.Windows.Forms.DataGridColumnStyle.MappingName** value to

the **System.Data.DataColumn.ColumnName** of a **System.Data.DataColumn** .

d) *DataGridBoolColumn*

*Example Syntax:*

e) *WndProc*

[C#] public DataGridBoolColumn(PropertyDescriptor prop);

[C++] public: DataGridBoolColumn(PropertyDescriptor\* prop);

[VB] Public Sub New(ByVal prop As PropertyDescriptor)

[JScript] public function DataGridBoolColumn(prop : PropertyDescriptor);

#### *Description*

Initializes a new instance of a **System.Windows.Forms.DataGridBoolColumn** with the specified **System.Data.DataColumn** .

The **System.Windows.Forms.DataGridBoolColumn** must be associated with a data source that contains Boolean values. The **System.ComponentModel.PropertyDescriptor** associated with the column.

f) *DataGridBoolColumn*

*Example Syntax:*

g) *WndProc*

[C#] public DataGridBoolColumn(PropertyDescriptor prop, bool isDefault);

[C++] public: DataGridBoolColumn(PropertyDescriptor\* prop, bool isDefault);

[VB] Public Sub New(ByVal prop As PropertyDescriptor, ByVal isDefault As Boolean)

[JScript] public function DataGridBoolColumn(prop : PropertyDescriptor,

isDefault : Boolean);

### *Description*

Initializes a new instance of a **System.Windows.Forms.DataGridBoolColumn** with the specified **System.ComponentModel.PropertyDescriptor** , and specifying whether the column style is a default column.

To get a **System.ComponentModel.PropertyDescriptor** , first use the **System.Windows.Forms.BindingContext** to return the appropriate **System.Windows.Forms.BindingManagerBase** object. Then use the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** to return a **System.ComponentModel.PropertyDescriptorCollection** . Finally, use the this property of the **System.ComponentModel.PropertyDescriptorCollection** to return the specific **System.ComponentModel.PropertyDescriptor** for the column. The **System.ComponentModel.PropertyDescriptor** associated with the column. **true** to specify the column as the default; otherwise, **false**.

*h) Alignment*

*i) AllowNull*

*j) WndProc*

### *Description*

Gets or sets a value indicating whether null values are allowed.

- k) *Container*
- l) *DataGridTableStyle*
- m) *DesignMode*
- n) *Events*
- o) *FalseValue*
- p) *WndProc*

#### *Description*

Gets or sets the actual value used when setting the value of the column to **false**.

- q) *FontHeight*
- r) *HeaderAccessibleObject*
- s) *HeaderText*
- t) *MappingName*
- u) *NullText*
- v) *NullValue*
- w) *WndProc*

#### *Description*

Gets or sets the actual value used when setting the value of the column to **System.DBNull.Value**.

You can also specify what text will be displayed by setting the **System.Windows.Forms.DataGridColumnStyle.NullText** property.



x) *PropertyDescriptor*

y) *ReadOnly*

z) *Site*

aa) *TrueValue*

bb) *WndProc*

*Description*

Gets or sets the actual value used when setting the value of the column to **true** .

cc) *Width*

dd) *WndProc*

*Description*

Occurs when the **System.Windows.Forms.DataGridBoolColumn.AllowNull** property is changed.

ee) *WndProc*

*Description*

Occurs when the **System.Windows.Forms.DataGridBoolColumn.FalseValue** property is changed.

*ff) WndProc*

*Description*

Occurs when the **System.Windows.Forms.DataGridBoolColumn.TrueValue** property value is changed.

For more information about handling events, see .

*gg) Abort*

[C#] protected internal override void Abort(int rowNum);

[C++] protected public: void Abort(int rowNum);

[VB] Overrides Protected Friend Dim Sub Abort(ByVal rowNum As Integer)

[JScript] package override function Abort(rowNum : int);

*Description*

Initiates a request to interrupt an edit procedure. The number of the row in which an operation is being interrupted.

*hh) Commit*

[C#] protected internal override bool Commit(CurrencyManager dataSource, int rowNum);

[C++] protected public: bool Commit(CurrencyManager\* dataSource, int rowNum);

[VB] Overrides Protected Friend Dim Function Commit(ByVal dataSource As CurrencyManager, ByVal rowNum As Integer) As Boolean

```

1 [JScript] package override function Commit(dataSource : CurrencyManager,
2 rowNum          :          int)          :          Boolean;
3

```

#### 4 *Description*

5 Initiates a request to complete an editing procedure.

6 *Return Value:* **true** if the editing procedure committed successfully; otherwise,  
7 **false** . The **System.Data.DataView** of the edited column. The number of the  
8 edited row.

#### 9 *ii) ConcedeFocus*

```

10 [C#]      protected      internal      override      void      ConcedeFocus();
11 [C++]      protected      public:      void      ConcedeFocus();
12 [VB]      Overrides      Protected      Friend      Dim      Sub      ConcedeFocus()
13 [JScript]      package      override      function      ConcedeFocus();
14

```

#### 15 *Description*

#### 16 *jj) Edit*

```

17
18
19 [C#] protected internal override void Edit(CurrencyManager source, int rowNum,
20 Rectangle bounds, bool readOnly, string instantText, bool cellIsVisible);
21 [C++] protected public: void Edit(CurrencyManager* source, int rowNum,
22 Rectangle bounds, bool readOnly, String* instantText, bool cellIsVisible);
23 [VB] Overrides Protected Friend Dim Sub Edit(ByVal source As
24 CurrencyManager, ByVal rowNum As Integer, ByVal bounds As Rectangle,
25 ByVal readOnly As Boolean, ByVal instantText As String, ByVal cellIsVisible As

```

Boolean)

[JScript] package override function Edit(source : CurrencyManager, rowNum : int, bounds : Rectangle, readOnly : Boolean, instantText : String, cellIsVisible : Boolean); Prepares the cell for editing a value.

### Description

Prepares the cell for editing a value.

Unlike the typical implementation of this method (as described in the **System.Windows.Forms.DataGridColumnStyle** class), the **System.Windows.Forms.DataGridBoolColumn.Edit(System.Windows.Forms.CurrencyManager, System.Int32, System.Drawing.Rectangle, System.Boolean, System.String, System.Boolean)** method doesn't site a control for editing the column value. Instead a check box is drawn when the **System.Windows.Forms.DataGridBoolColumn.Paint(System.Drawing.Graphics, System.Drawing.Rectangle, System.Windows.Forms.CurrencyManager, System.Int32)** method is called. The **System.Data.DataView** of the edited column. The row number of the edited column. The **System.Drawing.Rectangle** in which the control is to be sited. **true** if the value is read only; otherwise, **false**. The text to display in the cell. **true** to show the cell; otherwise, **false**.

### kk) EnterNullValue

[C#]       protected       internal       override       void       EnterNullValue();

[C++]       protected       public:       void       EnterNullValue();

[VB]       Overrides       Protected       Friend       Dim       Sub       EnterNullValue()

[JScript]       package       override       function       EnterNullValue();

### Description

Enters a **System.DBNull.Value** into the column.

To set the actual value used when entering a null value, use the **System.Windows.Forms.DataGridBoolColumn.NullValue** property.

## II) *GetColumnValueAtRow*

```
1  
2 [C#] protected internal override object GetColumnValueAtRow(CurrencyManager  
3 lm, int row);  
4 [C++] protected public: Object* GetColumnValueAtRow(CurrencyManager* lm,  
5 int row);  
6 [VB] Overrides Protected Friend Dim Function GetColumnValueAtRow(ByVal  
7 lm As CurrencyManager, ByVal row As Integer) As Object  
8 [JScript] package override function GetColumnValueAtRow(lm :  
9 CurrencyManager, row : int) : Object;
```

### *Description*

Gets the value at the specified row.

**Return Value:** The value, typed as **System.Object**. The **System.Windows.Forms.CurrencyManager** for the column. The row number.

## *mm) GetMinimumHeight*

```
17 [C#] protected internal override int GetMinimumHeight();  
18 [C++] protected public: int GetMinimumHeight();  
19 [VB] Overrides Protected Friend Dim Function GetMinimumHeight() As Integer  
20 [JScript] package override function GetMinimumHeight() : int;
```

### *Description*

Gets the height of a cell in a column.

**Return Value:** The height of the column. The default is 16.

## *nn) GetPreferredHeight*

[C#] protected internal override int GetPreferredHeight(Graphics g, object value);  
[C++] protected public: int GetPreferredHeight(Graphics\* g, Object\* value);  
[VB] Overrides Protected Friend Dim Function GetPreferredHeight(ByVal g As Graphics, ByVal value As Object) As Integer  
[JScript] package override function GetPreferredHeight(g : Graphics, value : Object) : int;

### *Description*

Gets the height used when resizing columns.

*Return Value:* The height used to automatically resize cells in a column. A Graphics that draws on the screen. An **System.Object** that contains the value to be drawn to the screen.

## *oo) GetPreferredSize*

[C#] protected internal override Size GetPreferredSize(Graphics g, object value);  
[C++] protected public: Size GetPreferredSize(Graphics\* g, Object\* value);  
[VB] Overrides Protected Friend Dim Function GetPreferredSize(ByVal g As Graphics, ByVal value As Object) As Size  
[JScript] package override function GetPreferredSize(g : Graphics, value : Object) : Size;

### *Description*

Gets the optimum width and height of a cell given a specific value to contain.

*Return Value:* A **System.Drawing.Size** that contains the drawing information for the cell.

The

**System.Windows.Forms.DataGridBoolColumn.GetPreferredSize(System.Drawing.Graphics, System.Object)** method allows you to resize the column based on the value displayed. For example, if a cell contains an especially large value, you can use the **System.Windows.Forms.DataGridBoolColumn.GetPreferredSize(System.Drawing.Graphics, System.Object)** method to return the optimum size of the cell based on the value. The optimum size takes into account not only the size of the string, but also the font used to display it. A **System.Drawing.Graphics** that draws the cell. The value that must fit in the cell.

*pp) Paint*

[C#] protected internal override void Paint(Graphics g, Rectangle bounds, CurrencyManager source, int rowNum);

[C++] protected public: void Paint(Graphics\* g, Rectangle bounds, CurrencyManager\* source, int rowNum);

[VB] Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As Integer)

[JScript] package override function Paint(g : Graphics, bounds : Rectangle, source : CurrencyManager, rowNum : int); Paints a column.

### *Description*

Draws the **System.Windows.Forms.DataGridBoolColumn** with the given **System.Drawing.Graphics** , **System.Drawing.Rectangle** and row number.

This method paints the background with the background color in the graphics object passed in. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the column. The number of the row referred to in the underlying data.

qq) *Paint*

```
[C#] protected internal override void Paint(Graphics g, Rectangle bounds,
CurrencyManager source, int rowNum, bool alignToRight);
[C++] protected public: void Paint(Graphics* g, Rectangle bounds,
CurrencyManager* source, int rowNum, bool alignToRight);
[VB] Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal
bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As
Integer, ByVal alignToRight As Boolean)
[JScript] package override function Paint(g : Graphics, bounds : Rectangle, source
: CurrencyManager, rowNum : int, alignToRight : Boolean);
```

*Description*

Draws the **System.Windows.Forms.DataGridBoolColumn** with the given **System.Drawing.Graphics**, **System.Drawing.Rectangle**, row number, and alignment settings. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the column. The number of the row in the underlying data table being referred to. A value indicating whether to align the content to the right.

rr) *Paint*

```
[C#] protected internal override void Paint(Graphics g, Rectangle bounds,
CurrencyManager source, int rowNum, Brush backBrush, Brush foreBrush, bool
alignToRight);
[C++] protected public: void Paint(Graphics* g, Rectangle bounds,
CurrencyManager* source, int rowNum, Brush* backBrush, Brush* foreBrush,
bool alignToRight);
```



```

1 [VB] Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal
2 bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As
3 Integer, ByVal backBrush As Brush, ByVal foreBrush As Brush, ByVal
4 alignToRight As Boolean)
5 [JScript] package override function Paint(g : Graphics, bounds : Rectangle, source
6 : CurrencyManager, rowNum : int, backBrush : Brush, foreBrush : Brush,
7 alignToRight : Boolean);

```

### *Description*

Draws the **System.Windows.Forms.DataGridBoolColumn** with the given **System.Drawing.Graphics**, **System.Drawing.Rectangle**, row number, **System.Drawing.Brush**, and **System.Drawing.Color**.

Paints the check box in the column. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the column. The number of the row in the underlying data table being referred to. A **System.Drawing.Brush** used to paint the background color. A **System.Drawing.Color** used to paint the foreground color. A value indicating whether to align the content to the right.

*ss) SetColumnValueAtRow*

```

18 [C#] protected internal override void SetColumnValueAtRow(CurrencyManager
19 lm, int row, object value);
20 [C++] protected public: void SetColumnValueAtRow(CurrencyManager* lm, int
21 row, Object* value);
22 [VB] Overrides Protected Friend Dim Sub SetColumnValueAtRow(ByVal lm As
23 CurrencyManager, ByVal row As Integer, ByVal value As Object)
24 [JScript] package override function SetColumnValueAtRow(lm :
25

```

CurrencyManager, row : int, value : Object);

### *Description*

Sets the value a a specified row. The **System.Windows.Forms.CurrencyManager** for the column. The row number. The value to set, typed as **System.Object**.

DataGridCell structure (System.Windows.Forms)

#### *a) UpdateUI*

### *Description*

Identifies a cell in the grid.

The **System.Windows.Forms.DataGridCell** can be used in conjunction with the **System.Windows.Forms.DataGrid** control's **System.Windows.Forms.DataGrid.CurrentCell** property to get or set the value of any cell. Setting the **System.Windows.Forms.DataGrid** control's **System.Windows.Forms.DataGrid.CurrentCell** property to a **System.Windows.Forms.DataGridCell** causes the focus to move to the cell specified by the **System.Windows.Forms.DataGridCell** .

#### *b) DataGridCell*

*Example Syntax:*

#### *c) UpdateUI*

[C#] public DataGridCell(int r, int c);

[C++] public: DataGridCell(int r, int c);

[VB] Public Sub New(ByVal r As Integer, ByVal c As Integer)

[JScript] public function DataGridCell(r : int, c : int);

## Description

Initializes a new instance of the **System.Windows.Forms.DataGridCell** class. The number of a row in the **System.Windows.Forms.DataGrid**. The number of a column in the **System.Windows.Forms.DataGrid**.

d) *ColumnNumber*

e) *UpdateUI*

[C#]        public        int        ColumnNumber        {get;        set;}

[C++]    public: \_\_property int get\_ColumnNumber();public: \_\_property void set\_ColumnNumber(int);

[VB]        Public        Property        ColumnNumber        As        Integer

[JScript] public function get ColumnNumber() : int;public function set ColumnNumber(int);

## Description

Gets or sets the number of a column in the **System.Windows.Forms.DataGrid** control.

You can use the **System.Windows.Forms.DataGridCell.ColumnNumber** to specify a **System.Data.DataColumn** object in the **System.Data.DataTable** associated with the **System.Windows.Forms.DataGrid** control.

f) *RowNumber*

g) *UpdateUI*

[C#]        public        int        RowNumber        {get;        set;}

[C++]    public: \_\_property int get\_RowNumber();public: \_\_property void set\_RowNumber(int);

```

1  [VB]      Public      Property      RowNumber      As      Integer
2  [JScript] public function get RowNumber() : int;public function set
3  RowNumber(int);

```

#### 5 *Description*

6 Gets or sets the number of a row in the **System.Windows.Forms.DataGrid** control.

7 You can use the **System.Windows.Forms.DataGridCell.RowNumber** value  
8 to specify a **System.Data.DataRow** object in the **System.Data.DataTable**  
9 associated with the **System.Windows.Forms.DataGrid** control.

#### 10 *h) Equals*

```

11 [C#]      public      override      bool      Equals(object      o);
12 [C++]      public:      bool      Equals(Object*      o);
13 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean
14 [JScript] public override function Equals(o : Object) : Boolean;
15

```

#### 16 *Description*

17 Gets a value indicating whether the **System.Windows.Forms.DataGridCell** is  
18 identical to a second **System.Windows.Forms.DataGridCell** .

19 *Return Value:* **true** if the second object is identical to the first; otherwise, **false** .  
An object you are to comparing.

#### 20 *i) GetHashCode*

```

21
22 [C#]      public      override      int      GetHashCode();
23 [C++]      public:      int      GetHashCode();
24 [VB] Overrides Public Function GetHashCode() As Integer
25

```

1 [JScript] public override function GetHashCode() : int;

2  
3 *Description*

4 Gets a hash value that can be added to a **System.Collections.Hashtable** .

5 *Return Value:* A number that uniquely identifies the

6 **System.Windows.Forms.DataGridCell** in a **System.Collections.Hashtable**

7 *j) ToString*

8  
9 [C#] public override string ToString();

10 [C++] public: String\* ToString();

11 [VB] Overrides Public Function ToString() As String

12 [JScript] public override function ToString() : String;

13  
14 *Description*

15 Gets the row number and column number of the cell.

16 *Return Value:* A string containing the row number and column number.

17 DataGridColumnStyle.DataGridColumnHeaderAccessibleObject class

18 (System.Windows.Forms)

19  
20  
21 *a) ToString*

22  
23 *Description*

24 Provides an implementation for an object that can be inspected by an  
25 accessibility application.

26 *b) DataGridColumnStyle.DataGridColumnHeaderAccessibleObject*

27 *Example Syntax:*

c) *ToString*

```
1
2                                     public
3 [C#]
4 DataGridColumnStyle.DataGridColumnHeaderAccessibleObject(DataGridColumn
5 nStyle                                     owner);
6 [C++] public: DataGridColumnHeaderAccessibleObject(DataGridColumnStyle*
7 owner);
8 [VB] Public Sub New(ByVal owner As DataGridColumnStyle)
9 [JScript]                                     public                                     function
10 DataGridColumnStyle.DataGridColumnHeaderAccessibleObject(owner :
11 DataGridColumnStyle);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.DataGridColumnStyle.DataGridColumnHeaderAccessibleObject** class. The **System.Windows.Forms.DataGridColumnStyle** that hosts the object.

d) *Bounds*

e) *ToString*

```
19 [C#] public override Rectangle Bounds {get;}
20 [C++] public: __property virtual Rectangle get_Bounds();
21 [VB] Overrides Public ReadOnly Property Bounds As Rectangle
22 [JScript] public function get Bounds() : Rectangle;
```

*Description*

Gets the bounding rectangle of a column.

*f) DefaultAction*

*g) Description*

*h) Help*

*i) KeyboardShortcut*

*j) Name*

*k) ToString*

#### *Description*

Gets the name of the column that owns the accessibility object.

*l) Owner*

*m) ToString*

[C#]           protected           DataGridColumnStyle           Owner           {get;}

[C++]   protected:   \_\_property   DataGridColumnStyle\*   get\_Owner();

[VB]   Protected   ReadOnly   Property   Owner   As   DataGridColumnStyle

[JScript]   protected   function   get   Owner()   :   DataGridColumnStyle;

#### *Description*

Gets the column style object that owns the accessibility object.

n) *Parent*

o) *ToString*

```
[C#]      public      override      AccessibleObject      Parent      {get;}
[C++]     public:     __property     virtual     AccessibleObject*     get_Parent();
[VB]      Overrides   Public   ReadOnly   Property   Parent   As   AccessibleObject
[JScript]  public     function     get     Parent()      :      AccessibleObject;
```

*Description*

Gets the parent accessibility object.

p) *Role*

q) *ToString*

```
[C#]      public      override      AccessibleRole      Role      {get;}
[C++]     public:     __property     virtual     AccessibleRole     get_Role();
[VB]      Overrides   Public   ReadOnly   Property   Role   As   AccessibleRole
[JScript]  public     function     get     Role()      :      AccessibleRole;
```

*Description*

Gets the role of the the accessibility object.

r) *State*

s) *Value*

t) *Navigate*

```
[C#] public override AccessibleObject Navigate(AccessibleNavigation navdir);
```



```

1 [C++] public: AccessibleObject* Navigate(AccessibleNavigation navdir);
2 [VB] Overrides Public Function Navigate(ByVal navdir As AccessibleNavigation)
3 As AccessibleObject
4 [JScript] public override function Navigate(navdir : AccessibleNavigation) :
5 AccessibleObject;

```

#### *Description*

Allows navigation to another object.

*Return Value:* An **System.Windows.Forms.AccessibleObject** specified by the *navdir* parameter. One of the **System.Windows.Forms.AccessibleNavigation** values.

DataGridColumnStyle class (System.Windows.Forms)

#### *a) UseStdAccessibleObjects*

#### *Description*

Specifies the appearance and text formatting and behavior of a **System.Windows.Forms.DataGrid** control column. This class is abstract.

The collection of **System.Windows.Forms.DataGridColumnStyle** objects (the **System.Windows.Forms.GridColumnStylesCollection** ) is accessed through the **System.Windows.Forms.DataGrid** control's **System.Windows.Forms.DataGrid.TableStyles** property.

#### *b) DataGridColumnStyle*

*Example Syntax:*

#### *c) UseStdAccessibleObjects*

|       |         |                        |
|-------|---------|------------------------|
| [C#]  | public  | DataGridColumnStyle(); |
| [C++] | public: | DataGridColumnStyle(); |

```

1 [VB] Public Sub New()
2 [JScript] public function DataGridColumnStyle(); Initializes a new instance of the
3 System.Windows.Forms.DataGridColumnStyle class.

```

#### *Description*

In a derived class, initializes a new instance of the **System.Windows.Forms.DataGridColumnStyle** class.

When you create an instance of a **System.Windows.Forms.DataGridColumnStyle** , the following properties are initialized.

d) *DataGridColumnStyle*

*Example Syntax:*

e) *UseStdAccessibleObjects*

```

13
14 [C#] public DataGridColumnStyle(PropertyDescriptor prop);
15 [C++] public: DataGridColumnStyle(PropertyDescriptor* prop);
16 [VB] Public Sub New(ByVal prop As PropertyDescriptor)
17 [JScript] public function DataGridColumnStyle(prop : PropertyDescriptor);

```

#### *Description*

Initializes a new instance of the **System.Windows.Forms.DataGridColumnStyle** class with the specified **System.ComponentModel.PropertyDescriptor** .

To create a new **System.Windows.Forms.DataGridColumnStyle** , you must first get the **System.Windows.Forms.CurrencyManager** for the data source of the table to which the column will be added. See the **CurrencyManager** and **BindingManager** for details on getting specific **CurrencyManager** objects for a form. A that **System.ComponentModel.PropertyDescriptor** that provides the attributes for the column.

1        *f) Alignment*

2        *g) UseStdAccessibleObjects*

3  
4 [C#]    public    virtual    HorizontalAlignment    Alignment    {get;    set;}

5 [C++]   public:   \_\_property   virtual   HorizontalAlignment   get\_Alignment();public:

6   \_\_property            virtual            void            set\_Alignment(HorizontalAlignment);

7 [VB]    Overridable   Public   Property   Alignment   As   HorizontalAlignment

8 [JScript]   public function get Alignment() : HorizontalAlignment;public function

9   set                                                                   Alignment(HorizontalAlignment);

10  
11        *Description*

12        Gets or sets the alignment of text in a column.

13        *h) Container*

14        *i) DataGridTableStyle*

15        *j) UseStdAccessibleObjects*

16  
17  
18        *Description*

19        Gets the **System.Windows.Forms.DataGridTableStyle** for the column.

1        *k)      DesignMode*

2        *l)      Events*

3        *m)      FontHeight*

4        *n)      UseStdAccessibleObjects*

5  
6  
7        *Description*

8        Gets the height of the column's font.

9        *o)      HeaderAccessibleObject*

10       *p)      UseStdAccessibleObjects*

11  
12       [C#]      public      AccessibleObject      HeaderAccessibleObject      {get;}

13       [C++]    public:   \_\_property   AccessibleObject\*   get\_HeaderAccessibleObject();

14       [VB]    Public ReadOnly Property HeaderAccessibleObject As AccessibleObject

15       [JScript]   public   function   get   HeaderAccessibleObject() :   AccessibleObject;

16  
17       *Description*

18       Gets the **System.Windows.Forms.AccessibleObject** for the column.

19       *q)      HeaderText*

20       *r)      UseStdAccessibleObjects*

21  
22       [C#]      public      virtual      string      HeaderText      {get;      set;}

23       [C++]    public:   \_\_property   virtual   String\*   get\_HeaderText();public:   \_\_property

24       virtual                                  void                                  set\_HeaderText(String\*);

```

1 [VB]      Overridable      Public      Property      HeaderText      As      String
2 [JScript] public function get HeaderText() : String;public function set
3 HeaderText(String);

```

#### *Description*

Gets or sets the text of the column header.

The **System.Windows.Forms.DataGridColumnStyle.HeaderText** property is typically used to display a caption that is different from the **System.Windows.Forms.DataGridColumnStyle.MappingName** value when the **System.Windows.Forms.DataGridColumnStyle.MappingName** value isn't easily understandable. For example, you may change the **System.Windows.Forms.DataGridColumnStyle.HeaderText** to "First Name" when the is "FName".

s) *MappingName*

t) *UseStdAccessibleObjects*

```

14 [C#]      public      string      MappingName      {get;      set;}
15 [C++] public: __property String* get_MappingName();public: __property void
16 set_MappingName(String*);
17 [VB]      Public      Property      MappingName      As      String
18 [JScript] public function get MappingName() : String;public function set
19 MappingName(String);

```

#### *Description*

Gets or sets the name used to map the column style to a data member.

The **System.Windows.Forms.DataGridColumnStyle.MappingName** property is usually set to the **System.Data.DataColumn.ColumnName** of a **System.Data.DataColumn**. Whenever the **System.Data.DataTable** containing the **System.Data.DataColumn** is displayed, the

**System.Windows.Forms.DataGridColumnStyle** with the same **System.Windows.Forms.DataGridColumnStyle.MappingName** will be used to display the data.

u) *NullText*

v) *UseStdAccessibleObjects*

[C#] public virtual string NullText {get; set;}

[C++] public: \_\_property virtual String\* get\_NullText();public: \_\_property virtual void set\_NullText(String\*);

[VB] Overridable Public Property NullText As String

[JScript] public function get NullText() : String;public function set NullText(String);

### *Description*

Gets or sets the text that is displayed when the column contains **null**.

The **System.Data.DataColumn** class's **System.Data.DataColumn.AllowDBNull** property determines if a column can contain null values.

w) *PropertyDescriptor*

x) *UseStdAccessibleObjects*

[C#] public virtual PropertyDescriptor PropertyDescriptor {get; set;}

[C++] public: \_\_property virtual PropertyDescriptor\* get\_PropertyDescriptor();public: \_\_property virtual void set\_PropertyDescriptor(PropertyDescriptor\*);

[VB] Overridable Public Property PropertyDescriptor As PropertyDescriptor

```

1 [JScript] public function get PropertyDescriptor() : PropertyDescriptor;public
2 function          set          PropertyDescriptor(PropertyDescriptor);
3

```

#### *Description*

Gets or sets the **System.ComponentModel.PropertyDescriptor** that determines the attributes of data displayed by the **System.Windows.Forms.DataGridColumnStyle** .

The **System.ComponentModel.PropertyDescriptor** for a column is set using the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** . See the **System.Windows.Forms.DataGridColumnStyle.#ctor** constructor for an example of using the **System.ComponentModel.PropertyDescriptor** to create a new **System.Windows.Forms.DataGridColumnStyle** object.

y) *ReadOnly*

z) *UseStdAccessibleObjects*

```

14 [C#]      public      virtual      bool      ReadOnly      {get;      set;}

```

```

15 [C++] public: __property virtual bool get_ReadOnly();public: __property virtual
16 void          set_ReadOnly(bool);

```

```

17 [VB]      Overridable      Public      Property      ReadOnly      As      Boolean

```

```

18 [JScript] public function get ReadOnly() : Boolean;public function set
19 ReadOnly(Boolean);
20

```

#### *Description*

Gets or sets a value indicating whether the data in the column can be edited.

Make a column read-only if it contains a primary key or if its value is generated automatically (as when the **System.Data.DataColumn** object's **System.Data.DataColumn.AutoIncrement** property is set to **true** ).

1        *aa)    Site*

2        *bb)    Width*

3        *cc)    UseStdAccessibleObjects*

4  
5  
6        *Description*

7        Gets or sets the width of the column.

8        *dd)    UseStdAccessibleObjects*

9  
10       [C#]        public        event        EventHandler        AlignmentChanged;

11       [C++]        public:        \_\_event        EventHandler\*        AlignmentChanged;

12       [VB]        Public        Event        AlignmentChanged        As        EventHandler

13  
14        *Description*

15        Occurs when the  
16        **System.Windows.Forms.DataGridColumnStyle.Alignment** property value  
17        changes.

18        For more information about handling events, see .

19        *ee)    UseStdAccessibleObjects*

20  
21        *Description*

22        Occurs when the column's font changes.

23        For more information about handling events, see .



*ff) UseStdAccessibleObjects*

|       |         |         |                   |                    |
|-------|---------|---------|-------------------|--------------------|
| [C#]  | public  | event   | EventHandler      | HeaderTextChanged; |
| [C++] | public: | __event | EventHandler*     | HeaderTextChanged; |
| [VB]  | Public  | Event   | HeaderTextChanged | As EventHandler    |

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.HeaderText** property value changes.

For more information about handling events, see .

*gg) UseStdAccessibleObjects*

|       |         |         |                    |                     |
|-------|---------|---------|--------------------|---------------------|
| [C#]  | public  | event   | EventHandler       | MappingNameChanged; |
| [C++] | public: | __event | EventHandler*      | MappingNameChanged; |
| [VB]  | Public  | Event   | MappingNameChanged | As EventHandler     |

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.MappingName** value changes.

For more information about handling events, see .

*hh) UseStdAccessibleObjects*

|       |         |         |                 |                  |
|-------|---------|---------|-----------------|------------------|
| [C#]  | public  | event   | EventHandler    | NullTextChanged; |
| [C++] | public: | __event | EventHandler*   | NullTextChanged; |
| [VB]  | Public  | Event   | NullTextChanged | As EventHandler  |

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.NullText** value changes.

For more information about handling events, see .

*ii) UseStdAccessibleObjects*

|       |         |         |                           |                            |
|-------|---------|---------|---------------------------|----------------------------|
| [C#]  | public  | event   | EventHandler              | PropertyDescriptorChanged; |
| [C++] | public: | __event | EventHandler*             | PropertyDescriptorChanged; |
| [VB]  | Public  | Event   | PropertyDescriptorChanged | As EventHandler            |

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.PropertyDescriptor** property value changes.

For more information about handling events, see .

*jj) UseStdAccessibleObjects*

|       |         |         |                 |                  |
|-------|---------|---------|-----------------|------------------|
| [C#]  | public  | event   | EventHandler    | ReadOnlyChanged; |
| [C++] | public: | __event | EventHandler*   | ReadOnlyChanged; |
| [VB]  | Public  | Event   | ReadOnlyChanged | As EventHandler  |

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.ReadOnly** property value changes.

For more information about handling events, see .

*kk) UseStdAccessibleObjects*

```
1
2 [C#]      public      event      EventHandler      WidthChanged;
3 [C++]     public:     __event     EventHandler*      WidthChanged;
4 [VB]      Public      Event      WidthChanged      As      EventHandler
5
```

*Description*

Occurs when the **System.Windows.Forms.DataGridColumnStyle.Width** property value changes.

For more information about handling events, see .

*ll) Abort*

```
10
11
12 [C#]      protected   internal   abstract   void      Abort(int      rowNum);
13 [C++]     protected   public:    virtual   void      Abort(int      rowNum) = 0;
14 [VB]      MustOverride Protected Friend Dim Sub Abort(ByVal rowNum As Integer)
15 [JScript] package    abstract   function   Abort(rowNum      :      int);
16
```

*Description*

When overridden in a derived class, initiates a request to interrupt an edit procedure.

The **System.Windows.Forms.DataGridColumnStyle** must end any editing operations before returning. Use the **System.Windows.Forms.DataGridColumnStyle.Abort(System.Int32)** method to accomplish this. The row number upon which an operation is being interrupted.

*mm) BeginUpdate*

```
23
24
25 [C#]      protected           void      BeginUpdate();
```

|           |            |          |                |
|-----------|------------|----------|----------------|
| [C++]     | protected: | void     | BeginUpdate(); |
| [VB]      | Protected  | Sub      | BeginUpdate()  |
| [JScript] | protected  | function | BeginUpdate(); |

#### *Description*

Suspends the painting of the column until the **System.Windows.Forms.DataGridColumnStyle.EndUpdate** method is called.

When many changes are made to the appearance of a **System.Windows.Forms.DataGrid** control (whether the changes are in the attributes of the column or the data displayed by the control), you should invoke the **System.Windows.Forms.DataGridColumnStyle.BeginUpdate** method to temporarily freeze the drawing of the control. This results in less distraction to the user, and a performance gain. After all updates have been made, invoke the **System.Windows.Forms.DataGridColumnStyle.EndUpdate** method to resume drawing of the control.

#### *nn) CheckValidDataSource*

|           |            |          |                                                      |
|-----------|------------|----------|------------------------------------------------------|
| [C#]      | protected  | void     | CheckValidDataSource(CurrencyManager value);         |
| [C++]     | protected: | void     | CheckValidDataSource(CurrencyManager* value);        |
| [VB]      | Protected  | Sub      | CheckValidDataSource(ByVal value As CurrencyManager) |
| [JScript] | protected  | function | CheckValidDataSource(value : CurrencyManager);       |

#### *Description*

Throws an exception if the **System.Windows.Forms.DataGrid** does not have a valid data source, or if this column is not mapped to a valid property in the data source. A **System.Windows.Forms.CurrencyManager** to check.

#### *oo) ColumnStartedEditing*

|      |           |          |         |      |                              |
|------|-----------|----------|---------|------|------------------------------|
| [C#] | protected | internal | virtual | void | ColumnStartedEditing(Control |
|------|-----------|----------|---------|------|------------------------------|

```

1 editingControl);
2 [C++] protected public: virtual void ColumnStartedEditing(Control*
3 editingControl);
4 [VB] Overridable Protected Friend Dim Sub ColumnStartedEditing(ByVal
5 editingControl As Control)
6 [JScript] package function ColumnStartedEditing(editingControl : Control);
7

```

#### *Description*

1 Informs the **System.Windows.Forms.DataGrid** that the user has begun editing the column.

10 When called, the **System.Windows.Forms.IDataGridColumnStyleEditingNotificationService.ColumnStartedEditing(System.Windows.Forms.Control)** method  
11 allows the **System.Windows.Forms.DataGrid** control to show a pencil in the row header indicating the row is being edited. The  
12 **System.Windows.Forms.Control** that is editing the column.

#### *pp) Commit*

```

16 [C#] protected internal abstract bool Commit(CurrencyManager dataSource, int
17 rowNum);
18 [C++] protected public: virtual bool Commit(CurrencyManager* dataSource, int
19 rowNum) = 0;
20 [VB] MustOverride Protected Friend Dim Function Commit(ByVal dataSource As
21 CurrencyManager, ByVal rowNum As Integer) As Boolean
22 [JScript] package abstract function Commit(dataSource : CurrencyManager,
23 rowNum : int) : Boolean;
24
25

```

## Description

When overridden in a derived class, initiates a request to complete an editing procedure.

**Return Value:** **true** if the editing procedure committed successfully; otherwise, **false** .

Call the **System.Windows.Forms.DataGridColumnStyle.Commit(System.Windows.Forms.CurrencyManager,System.Int32)** method when the **System.Windows.Forms.DataGridColumnStyle** receives a request to complete editing. If this is not possible without error, return **false** . The **System.Windows.Forms.CurrencyManager** for the **System.Windows.Forms.DataGridColumnStyle**. The number of the row being edited.

### qq) *ConcedeFocus*

|           |             |           |         |          |                 |
|-----------|-------------|-----------|---------|----------|-----------------|
| [C#]      | protected   | internal  | virtual | void     | ConcedeFocus(); |
| [C++]     | protected   | public:   | virtual | void     | ConcedeFocus(); |
| [VB]      | Overridable | Protected | Friend  | Dim Sub  | ConcedeFocus()  |
| [JScript] |             | package   |         | function | ConcedeFocus(); |

## Description

Notifies a column that it has lost focus.

Use this method to determine a further action in a derived class. For example, this method is overridden by the **System.Windows.Forms.DataGridTextBoxColumn** to hide the **System.Windows.Forms.DataGridTextBoxColumn** .

### rr) *CreateHeaderAccessibleObject*

|      |           |         |                  |                                 |
|------|-----------|---------|------------------|---------------------------------|
| [C#] | protected | virtual | AccessibleObject | CreateHeaderAccessibleObject(); |
|------|-----------|---------|------------------|---------------------------------|

```

1 [C++] protected: virtual AccessibleObject* CreateHeaderAccessibleObject();
2 [VB] Overridable Protected Function CreateHeaderAccessibleObject() As
3 AccessibleObject
4 [JScript] protected function CreateHeaderAccessibleObject() : AccessibleObject;

```

#### *Description*

Gets the **System.Windows.Forms.AccessibleObject** for the column.  
*Return Value:* An **System.Windows.Forms.AccessibleObject** for the column.

This property is called by a **System.Windows.Forms.DataGridTableStyle** when its **System.Windows.Forms.DataGrid** property changes. This property is required because any child controls hosted by a **System.Windows.Forms.DataGridColumnStyle** must be added to the **System.Windows.Forms.DataGrid** control's **ControlCollection**.

#### *ss) Edit*

```

14 [C#] protected internal virtual void Edit(CurrencyManager source, int rowNum,
15 Rectangle bounds, bool readOnly);
16 [C++] protected public: virtual void Edit(CurrencyManager* source, int rowNum,
17 Rectangle bounds, bool readOnly);
18 [VB] Overridable Protected Friend Dim Sub Edit(ByVal source As
19 CurrencyManager, ByVal rowNum As Integer, ByVal bounds As Rectangle,
20 ByVal readOnly As Boolean)
21 [JScript] package function Edit(source : CurrencyManager, rowNum : int, bounds
22 : Rectangle, readOnly : Boolean); Prepares the cell for editing a value.

```

#### *Description*

Prepares a cell for editing.

Typically, the

**System.Windows.Forms.DataGridColumnStyle.Edit(System.Windows.Forms.CurrencyManager, System.Int32, System.Drawing.Rectangle, System.Boolean)** method sites a control onto the grid at the location of the cell being edited. The **System.Windows.Forms.CurrencyManager** for the **System.Windows.Forms.DataGridColumnStyle**. The row number to edit. The bounding **System.Drawing.Rectangle** in which the control is to be sited. A value indicating whether the column is a read-only.

*tt) Edit*

[C#] protected internal virtual void Edit(CurrencyManager source, int rowNum, Rectangle bounds, bool readOnly, string instantText);

[C++] protected public: virtual void Edit(CurrencyManager\* source, int rowNum, Rectangle bounds, bool readOnly, String\* instantText);

[VB] Overridable Protected Friend Dim Sub Edit(ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal bounds As Rectangle, ByVal readOnly As Boolean, ByVal instantText As String)

[JScript] package function Edit(source : CurrencyManager, rowNum : int, bounds : Rectangle, readOnly : Boolean, instantText : String);

*Description*

Prepares the cell for editing using the specified **System.Windows.Forms.CurrencyManager**, row number, and **System.Drawing.Rectangle** parameters.

Typically, the

**System.Windows.Forms.DataGridColumnStyle.Edit(System.Windows.Forms.CurrencyManager, System.Int32, System.Drawing.Rectangle, System.Boolean)** method sites a control onto the grid at the location of the cell being edited. The **System.Windows.Forms.CurrencyManager** for the **System.Windows.Forms.DataGridColumnStyle**. The row number in this column which is being edited. The **System.Drawing.Rectangle** in which the control is to be sited. A value indicating whether the column is a read-only. The text to display in the control.



uu) *Edit*

[C#] protected internal abstract void Edit(CurrencyManager source, int rowNum, Rectangle bounds, bool readOnly, string instantText, bool cellsVisible);

[C++] protected public: virtual void Edit(CurrencyManager\* source, int rowNum, Rectangle bounds, bool readOnly, String\* instantText, bool cellsVisible) = 0;

[VB] MustOverride Protected Friend Dim Sub Edit(ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal bounds As Rectangle, ByVal readOnly As Boolean, ByVal instantText As String, ByVal cellsVisible As Boolean)

[JScript] package abstract function Edit(source : CurrencyManager, rowNum : int, bounds : Rectangle, readOnly : Boolean, instantText : String, cellsVisible : Boolean); Prepares a cell for editing.

*Description*

When overridden in a deriving class, prepares a cell for editing.

Typically, the

**System.Windows.Forms.DataGridColumnStyle.Edit(System.Windows.Forms.CurrencyManager, System.Int32, System.Drawing.Rectangle, System.Boolean)** method sites a control onto the grid at the location of the cell being edited. The **System.Windows.Forms.CurrencyManager** for the **System.Windows.Forms.DataGridColumnStyle**. The row number in this column which is being edited. The **System.Drawing.Rectangle** in which the control is to be sited. A value indicating whether the column is a read-only. The text to display in the control. A value indicating whether the cell is visible.

vv) *EndUpdate*

[C#] protected void EndUpdate();

[C++] protected: void EndUpdate();

|           |           |          |              |
|-----------|-----------|----------|--------------|
| [VB]      | Protected | Sub      | EndUpdate()  |
| [JScript] | protected | function | EndUpdate(); |

#### Description

Resumes the painting of columns suspended by calling the **System.Windows.Forms.DataGridColumnStyle.BeginUpdate** method.

When many changes are made to the appearance of a **System.Windows.Forms.DataGrid** control (whether the changes are in the attributes of the column or the data displayed by the control), you should invoke the **System.Windows.Forms.DataGridColumnStyle.BeginUpdate** method to temporarily freeze the drawing of the control. This results in less distraction to the user, and a performance gain. After all updates have been made, invoke the **System.Windows.Forms.DataGridColumnStyle.EndUpdate** method to resume drawing of the control.

*ww) EnterNullValue*

|           |             |           |          |         |                   |
|-----------|-------------|-----------|----------|---------|-------------------|
| [C#]      | protected   | internal  | virtual  | void    | EnterNullValue(); |
| [C++]     | protected   | public:   | virtual  | void    | EnterNullValue(); |
| [VB]      | Overridable | Protected | Friend   | Dim Sub | EnterNullValue()  |
| [JScript] | package     |           | function |         | EnterNullValue(); |

#### Description

Enters a **System.DBNull.Value** into the column.

This method is called when the user presses Alt+0 to allow a column to enter the appropriate null value. For example, when called on a **System.Windows.Forms.DataGridTextBoxColumn**, the appropriate **System.Windows.Forms.DataGridColumnStyle.NullText** value is inserted into the column.

xx) *GetColumnValueAtRow*

```
1  
2 [C#] protected internal virtual object GetColumnValueAtRow(CurrencyManager  
3 source, int rowNum);  
4 [C++] protected public: virtual Object*  
5 GetColumnValueAtRow(CurrencyManager* source, int rowNum);  
6 [VB] Overridable Protected Friend Dim Function GetColumnValueAtRow(ByVal  
7 source As CurrencyManager, ByVal rowNum As Integer) As Object  
8 [JScript] package function GetColumnValueAtRow(source : CurrencyManager,  
9 rowNum : int) : Object;
```

11 *Description*

12 Gets the value in the specified row from the specified  
13 **System.Windows.Forms.CurrencyManager** .  
14 *Return Value:* An **System.Object** containing the value.

15 If the data source for the column is a **System.Data.DataTable** , use the  
16 **System.Data.DataTable.ColumnChanging** or  
17 **System.Data.DataTable.RowChanging** events to determine when a row or  
18 column value has changed. The **System.Windows.Forms.CurrencyManager**  
19 containing the data. The row number containing the data.

yy) *GetMinimumHeight*

```
20 [C#] protected internal abstract int GetMinimumHeight();  
21 [C++] protected public: virtual int GetMinimumHeight() = 0;  
22 [VB] MustOverride Protected Friend Dim Function GetMinimumHeight() As  
23 Integer  
24 [JScript] package abstract function GetMinimumHeight() : int;
```

1  
2 *Description*

3 When overridden in a derived class, gets the minimum height of a row.

4 *Return Value:* The minimum height of a row.

5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
*zz) GetPreferredHeight*

[C#] protected internal abstract int GetPreferredHeight(Graphics g, object value);

[C++] protected public: virtual int GetPreferredHeight(Graphics\* g, Object\* value) = 0;

[VB] MustOverride Protected Friend Dim Function GetPreferredHeight(ByVal g As Graphics, ByVal value As Object) As Integer

[JScript] package abstract function GetPreferredHeight(g : Graphics, value : Object) : int;

14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
*Description*

15 When overridden in a derived class, gets the height used for automatically  
16 resizing columns.

17 *Return Value:* The height used for auto resizing a cell.

18 The  
19 **System.Windows.Forms.DataGridColumnStyle.GetPreferredSize(System.Drawing.Graphics, System.Object)** will usually be invoked from a mouse  
20 down event to resize a column's height for a long string. A  
21 **System.Drawing.Graphics** object. A object value for which you want to know  
22 the screen height and width.

21  
22  
23  
24  
25  
*aaa) GetPreferredSize*

[C#] protected internal abstract Size GetPreferredSize(Graphics g, object value);

[C++] protected public: virtual Size GetPreferredSize(Graphics\* g, Object\* value)

```

1 = 0;
2 [VB] MustOverride Protected Friend Dim Function GetPreferredSize(ByVal g As
3 Graphics, ByVal value As Object) As Size
4 [JScript] package abstract function GetPreferredSize(g : Graphics, value : Object)
5 : Size;
6

```

#### Description

When overridden in a derived class, gets the width and height of the specified value. The width and height are used when the user navigates to

**System.Windows.Forms.DataGridTableStyle** using the **System.Windows.Forms.DataGridColumnStyle**.

*Return Value:* A **System.Drawing.Size** that contains the dimensions of the cell.

#### Use

**System.Windows.Forms.DataGridColumnStyle.GetPreferredSize(System.Drawing.Graphics, System.Object)** to determine the width a column should resize to, given a particular string or numeral. A

**System.Drawing.Graphics** object. An object value for which you want to know the screen height and width.

#### bbb) Invalidate

|           |             |           |          |               |
|-----------|-------------|-----------|----------|---------------|
| [C#]      | protected   | virtual   | void     | Invalidate(); |
| [C++]     | protected:  | virtual   | void     | Invalidate(); |
| [VB]      | Overridable | Protected | Sub      | Invalidate()  |
| [JScript] | protected   |           | function | Invalidate(); |

#### Description

Redraws the column and causes a paint message to be sent to the control.

The **System.Windows.Forms.DataGridColumnStyle.Invalidate** method is typically called after an editing operation is interrupted. For example, you may call the method when implementing the

**System.Windows.Forms.DataGridColumnStyle.Abort(System.Int32)**  
method.

*ccc) Paint*

[C#] protected internal abstract void Paint(Graphics g, Rectangle bounds, CurrencyManager source, int rowNum);

[C++] protected public: virtual void Paint(Graphics\* g, Rectangle bounds, CurrencyManager\* source, int rowNum) = 0;

[VB] MustOverride Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As Integer)

[JScript] package abstract function Paint(g : Graphics, bounds : Rectangle, source : CurrencyManager, rowNum : int); When overridden in a derived class, paints the column in a **System.Windows.Forms.DataGrid** control.

#### *Description*

Paints the a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics** , **System.Drawing.Rectangle** , **System.Windows.Forms.CurrencyManager** , and row number.

This method will be called very often with state = **DataGridColumnStyleState.Normal** , so inheriting classes should heavily optimize that type of call. When painting a cell, keep in mind the following: the border painting logic is handled elsewhere. Callers should save all Pens, Brushes, etc. before passing their **System.Drawing.Graphics** object to this method. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** control the column belongs to. The number of the row in the underlying data being referred to.

### *ddd) Paint*

```
1
2 [C#] protected internal abstract void Paint(Graphics g, Rectangle bounds,
3 CurrencyManager source, int rowNum, bool alignToRight);
4
5 [C++] protected public: virtual void Paint(Graphics* g, Rectangle bounds,
6 CurrencyManager* source, int rowNum, bool alignToRight) = 0;
7
8 [VB] MustOverride Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal
9 bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As
10 Integer, ByVal alignToRight As Boolean)
11
12 [JScript] package abstract function Paint(g : Graphics, bounds : Rectangle, source
13 : CurrencyManager, rowNum : int, alignToRight : Boolean);
```

### *Description*

When overridden in a derived class, paints a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics**, **System.Drawing.Rectangle**, **System.Windows.Forms.CurrencyManager**, row number, and alignment. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** control the column belongs to. The number of the row in the underlying data being referred to. A value indicating whether to align the column's content to the right.

### *eee) Paint*

```
21
22 [C#] protected internal virtual void Paint(Graphics g, Rectangle bounds,
23 CurrencyManager source, int rowNum, Brush backBrush, Brush foreBrush, bool
24 alignToRight);
25
26 [C++] protected public: virtual void Paint(Graphics* g, Rectangle bounds,
```

```

1 CurrencyManager* source, int rowNum, Brush* backBrush, Brush* foreBrush,
2 bool alignToRight);
3 [VB] Overridable Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal
4 bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As
5 Integer, ByVal backBrush As Brush, ByVal foreBrush As Brush, ByVal
6 alignToRight As Boolean)
7 [JScript] package function Paint(g : Graphics, bounds : Rectangle, source :
8 CurrencyManager, rowNum : int, backBrush : Brush, foreBrush : Brush,
9 alignToRight : Boolean);

```

#### *Description*

Paints a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics**, **System.Drawing.Rectangle**, **System.Windows.Forms.CurrencyManager**, row number, background color, foreground color, and alignment. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** control the column belongs to. The number of the row in the underlying data table being referred to. A **System.Drawing.Brush** used to paint the background color. A **System.Drawing.Color** used to paint the foreground color. A value indicating whether to align the content to the right.

#### *fff) ResetHeaderText*

|           |         |          |                    |
|-----------|---------|----------|--------------------|
| [C#]      | public  | void     | ResetHeaderText(); |
| [C++]     | public: | void     | ResetHeaderText(); |
| [VB]      | Public  | Sub      | ResetHeaderText()  |
| [JScript] | public  | function | ResetHeaderText(); |



## Description

Resets the **System.Windows.Forms.DataGridColumnStyle.HeaderText** to its default value, **null**.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGrid**, or creating your own control incorporating the **System.Windows.Forms.DataGrid**.

### ggg) SetColumnValueAtRow

[C#] protected internal virtual void SetColumnValueAtRow(CurrencyManager source, int rowNum, object value);

[C++] protected public: virtual void SetColumnValueAtRow(CurrencyManager\* source, int rowNum, Object\* value);

[VB] Overridable Protected Friend Dim Sub SetColumnValueAtRow(ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal value As Object)

[JScript] package function SetColumnValueAtRow(source : CurrencyManager, rowNum : int, value : Object);

## Description

Sets the value in a specified row with the value from a specified **System.Windows.Forms.CurrencyManager**. A **System.Windows.Forms.CurrencyManager** associated with the **System.Windows.Forms.DataGridColumnStyle**. The number of the row. The value to set.

### hhh) SetDataGrid

[C#] protected virtual void SetDataGrid(DataGrid value);

[C++] protected: virtual void SetDataGrid(DataGrid\* value);

1 [VB] Overridable Protected Sub SetDataGrid(ByVal value As DataGrid)

2 [JScript] protected function SetDataGrid(value : DataGrid);

3  
4 *Description*

5 Sets the **System.Windows.Forms.DataGrid** control that this column belongs to.

6 Typically, you should use the  
7 **System.Windows.Forms.DataGridColumnStyle.SetDataGridInColumn(System.Windows.Forms.DataGrid)** method. The  
8 **System.Windows.Forms.DataGrid** control that this column belongs to.

9 *iii) SetDataGridInColumn*

10  
11 [C#] protected virtual void SetDataGridInColumn(DataGrid value);

12 [C++] protected: virtual void SetDataGridInColumn(DataGrid\* value);

13 [VB] Overridable Protected Sub SetDataGridInColumn(ByVal value As  
14 DataGrid)

15 [JScript] protected function SetDataGridInColumn(value : DataGrid);

16  
17 *Description*

18 Sets the **System.Windows.Forms.DataGrid** for the column.

19 This method is typically overridden by derived classes to do special processing  
20 when the **System.Windows.Forms.DataGridColumnStyle** is added to  
21 **System.Windows.Forms.DataGrid** . For example, the  
22 **System.Windows.Forms.DataGridTextBoxColumn** uses this method to add  
23 the **System.Windows.Forms.DataGridTextBoxColumn** as a child of the  
24 **System.Windows.Forms.DataGrid** . A **System.Windows.Forms.DataGrid**.  
25



*Description*

Updates the value of a specified row with the given text. The **System.Windows.Forms.CurrencyManager** associated with the **System.Windows.Forms.DataGridColumnStyle**. The row to update. The new value.

DataGridLineStyle enumeration (System.Windows.Forms)

*a) UpdateUI*

*Description*

Specifies the style of gridlines in a **System.Windows.Forms.DataGrid** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.DataGrid.GridLineStyle** property in the **System.Windows.Forms.DataGrid** class. The default is **Solid** .

*b) UpdateUI*

|           |         |       |                   |                      |
|-----------|---------|-------|-------------------|----------------------|
| [C#]      | public  | const | DataGridLineStyle | None;                |
| [C++]     | public: | const | DataGridLineStyle | None;                |
| [VB]      | Public  | Const | None              | As DataGridLineStyle |
| [JScript] | public  | var   | None              | : DataGridLineStyle; |

*Description*

No gridlines between cells.

*c) UpdateUI*

|      |        |       |                   |        |
|------|--------|-------|-------------------|--------|
| [C#] | public | const | DataGridLineStyle | Solid; |
|------|--------|-------|-------------------|--------|

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C++]     | public: | const | DataGridLineStyle | Solid;             |
| [VB]      | Public  | Const | Solid As          | DataGridLineStyle  |
| [JScript] | public  | var   | Solid :           | DataGridLineStyle; |

*Description*

Solid gridlines between cells.

DataGridParentRowsLabelStyle enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies how the parent row labels of a **System.Windows.Forms.DataGrid** control are displayed.

Use the members of this enumeration to set the value of the **System.Windows.Forms.DataGrid.ParentRowsLabelStyle** property.

*b) ToString*

|           |         |       |                              |                               |
|-----------|---------|-------|------------------------------|-------------------------------|
| [C#]      | public  | const | DataGridParentRowsLabelStyle | Both;                         |
| [C++]     | public: | const | DataGridParentRowsLabelStyle | Both;                         |
| [VB]      | Public  | Const | Both As                      | DataGridParentRowsLabelStyle  |
| [JScript] | public  | var   | Both :                       | DataGridParentRowsLabelStyle; |

*Description*

Displays both the parent table and column names.

c) *ToString*

```
[C#]    public    const    DataGridParentRowsLabelStyle    ColumnName;  
[C++]   public:   const    DataGridParentRowsLabelStyle    ColumnName;  
[VB]    Public    Const    ColumnName    As    DataGridParentRowsLabelStyle  
[JScript] public var    ColumnName    :    DataGridParentRowsLabelStyle;
```

*Description*

Displays the parent column name.

d) *ToString*

```
[C#]    public    const    DataGridParentRowsLabelStyle    None;  
[C++]   public:   const    DataGridParentRowsLabelStyle    None;  
[VB]    Public    Const    None    As    DataGridParentRowsLabelStyle  
[JScript] public var    None    :    DataGridParentRowsLabelStyle;
```

*Description*

Display no parent row labels.

e) *ToString*

```
[C#]    public    const    DataGridParentRowsLabelStyle    TableName;  
[C++]   public:   const    DataGridParentRowsLabelStyle    TableName;  
[VB]    Public    Const    TableName    As    DataGridParentRowsLabelStyle  
[JScript] public var    TableName    :    DataGridParentRowsLabelStyle;
```

*Description*

Displays the parent table name.

**DataGridPreferredColumnWidthTypeConverter** class

(System.Windows.Forms)

*a) ToString*

*Description*

Converts the value of an object to a different data type.

*b) DataGridPreferredColumnWidthTypeConverter*

*Example Syntax:*

*c) ToString*

[C#] public DataGridPreferredColumnWidthTypeConverter();

[C++] public: DataGridPreferredColumnWidthTypeConverter();

[VB] Public Sub New()

[JScript] public function DataGridPreferredColumnWidthTypeConverter();

*d) CanConvertFrom*

[C#] public override bool CanConvertFrom(ITypeDescriptorContext context,  
Type sourceType);

[C++] public: bool CanConvertFrom(ITypeDescriptorContext\* context, Type\*  
sourceType);

```

1 [VB] Overrides Public Function CanConvertFrom(ByVal context As
2 ITypeDescriptorContext, ByVal sourceType As Type) As Boolean
3 [JScript] public override function CanConvertFrom(context :
4 ITypeDescriptorContext, sourceType : Type) : Boolean;

```

#### *Description*

Gets a value that specifies whether the converter can convert an object in the given source type to the native type of the converter.

*Return Value:* **true** , if this converter can perform the conversion; otherwise, false. An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you want to convert from.

#### *e) ConvertFrom*

```

13 [C#] public override object ConvertFrom(ITypeDescriptorContext context,
14 CultureInfo culture, object value);
15 [C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
16 CultureInfo* culture, Object* value);
17 [VB] Overrides Public Function ConvertFrom(ByVal context As
18 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
19 As Object
20 [JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
21 culture : CultureInfo, value : Object) : Object;

```

#### *Description*

Converts the given object to the native type of the converter. An **System.ComponentModel.ITypeDescriptorContext** that a provides format



context. A **System.Globalization.CultureInfo** to use for the current culture.  
The **System.Object** to convert.

*f) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
culture, Object* value, Type* destinationType);
[VB] Overrides Public Function ConvertTo(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
ByVal destinationType As Type) As Object
[JScript] public override function ConvertTo(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object, destinationType : Type) : Object;
```

*Description*

Converts the given value to the native type of the converter. An **System.ComponentModel.ITypeDescriptorContext** that provides format context. A **System.Globalization.CultureInfo** to use for the current culture. The **System.Object** to convert. The **System.Type** of the conversion.

DataGridTableStyle class (System.Windows.Forms)

*a) ToString*

*Description*

Represents the table drawn by the **System.Windows.Forms.DataGrid** control at run time.

The **System.Windows.Forms.DataGrid** control displays data in the form of a grid. The **System.Windows.Forms.DataGridTableStyle** is a class that

represents the drawn grid only. This grid should not be confused with the **System.Data.DataTable** class, which is a possible source of data for the grid. Instead, the **System.Data.DataGridTableStyle** strictly represents the grid as it is painted in the control. Therefore, through the **System.Windows.Forms.DataGridTableStyle** you can control the appearance of the grid for each **System.Data.DataTable**. To specify which **System.Windows.Forms.DataGridTableStyle** is used when displaying data from a particular **System.Data.DataTable**, set the **System.Windows.Forms.DataGridTableStyle.MappingName** to the **System.Data.DataTable.TableName** of a **System.Data.DataTable**.

#### b) ToString

```
[C#]      public      static      DataGridTableStyle      DefaultTableStyle;
[C++]     public:     static      DataGridTableStyle*      DefaultTableStyle;
[VB]      Public     Shared      DefaultTableStyle      As      DataGridTableStyle
[JScript] public     static     var      DefaultTableStyle :      DataGridTableStyle;
```

#### Description

Gets the default table style.

#### c) DataGridTableStyle

*Example Syntax:*

#### d) ToString

```
[C#]      public      DataGridTableStyle();
[C++]     public:     DataGridTableStyle();
[VB]      Public     Sub      New()
[JScript] public function DataGridTableStyle(); Initializes a new instance of the
System.Windows.Forms.DataGridTableStyle class.
```

1  
2 *Description*

3 Initializes a new instance of the  
4 **System.Windows.Forms.DataGridTableStyle** class.

5 *e) DataGridTableStyle*

6 *Example Syntax:*

7 *f) ToString*

8 [C#] public DataGridTableStyle(bool isDefaultTableStyle);

9 [C++] public: DataGridTableStyle(bool isDefaultTableStyle);

10 [VB] Public Sub New(ByVal isDefaultTableStyle As Boolean)

11 [JScript] public function DataGridTableStyle(isDefaultTableStyle : Boolean);

12  
13  
14 *Description*

15 Initializes a new instance of the  
16 **System.Windows.Forms.DataGridTableStyle** class using the specified value  
17 to determine whether the grid table is the default style. **true** to specify the table  
18 as the default; otherwise, **false**.

19 *g) DataGridTableStyle*

20 *Example Syntax:*

21 *h) ToString*

22 [C#] public DataGridTableStyle(CurrencyManager listManager);

23 [C++] public: DataGridTableStyle(CurrencyManager\* listManager);

24 [VB] Public Sub New(ByVal listManager As CurrencyManager)

25 [JScript] public function DataGridTableStyle(listManager : CurrencyManager);

1  
2 *Description*

3 Initializes a new instance of the  
4 **System.Windows.Forms.DataGridTableStyle** class with the specified  
5 **System.Windows.Forms.CurrencyManager** . The  
6 **System.Windows.Forms.CurrencyManager** to use.

7  
8  
9 i) *AllowSorting*

10 j) *ToString*

11  
12 [C#] public bool AllowSorting {get; set;}

13 [C++] public: \_\_property bool get\_AllowSorting();public: \_\_property void  
14 set\_AllowSorting(bool);

15 [VB] Public Property AllowSorting As Boolean

16 [JScript] public function get AllowSorting() : Boolean;public function set  
17 AllowSorting(Boolean);

18  
19 *Description*

20 Indicates whether sorting is allowed on the grid table when this  
21 **System.Windows.Forms.DataGridTableStyle** is used.

22 When the **System.Windows.Forms.DataGridTableStyle.AllowSorting**  
23 property is set to **true** , a triangle appears in each column header indicating the  
24 direction of the sort. The user can click on any column header to sort the grid by  
25 that column. Clicking the column a second time changes the direction of the sort.

26  
27 k) *AlternatingBackColor*

28 l) *ToString*

29 [C#] public Color AlternatingBackColor {get; set;}

30 [C++] public: \_\_property Color get\_AlternatingBackColor();public: \_\_property

```

1 void set_AlternatingBackColor(Color);
2 [VB] Public Property AlternatingBackColor As Color
3 [JScript] public function get AlternatingBackColor() : Color;public function set
4 AlternatingBackColor(Color);

```

#### *Description*

The background color of alternating rows.

*m) BackColor*

*n) ToString*

```

11 [C#] public Color BackColor {get; set;}
12 [C++] public: __property Color get_BackColor();public: __property void
13 set_BackColor(Color);
14 [VB] Public Property BackColor As Color
15 [JScript] public function get BackColor() : Color;public function set
16 BackColor(Color);

```

#### *Description*

Gets or sets the background color of the grid table.

The **System.Windows.Forms.DataGridTableStyle.AlternatingBackColor** property can also be set to create a ledger-like appearance.

*o) ColumnHeadersVisible*

*p) ToString*

```

24
25 [C#] public bool ColumnHeadersVisible {get; set;}

```

```

1 [C++] public: __property bool get_ColumnHeadersVisible();public: __property
2 void set_ColumnHeadersVisible(bool);
3 [VB] Public Property ColumnHeadersVisible As Boolean
4 [JScript] public function get ColumnHeadersVisible() : Boolean;public function
5 set ColumnHeadersVisible(Boolean);
6

```

### *Description*

Gets or sets a value indicating whether column headers are visible.

To set header caption text, use the **System.Windows.Forms.DataGridColumnStyle.HeaderText** property of the **System.Windows.Forms.DataGridColumnStyle** class.

*q) Container*

*r) DataGrid*

*s) ToString*

### *Description*

Gets or sets the **System.Windows.Forms.DataGrid** control for the drawn table.

The **System.Windows.Forms.DataGrid** control displays data in the form of a grid. The **System.Windows.Forms.DataGridTableStyle** is an object that represents the drawn grid. The **System.Windows.Forms.DataGrid** property returns a reference to the control that is displaying the grid.

1            *t)      **DesignMode***

2            *u)      **Events***

3            *v)      **ForeColor***

4            *w)      **ToString***

5  
6  
7            *Description*

8            Gets or sets the foreground color of the grid table.

9            *x)      **GridColumnStyles***

10           *y)      **ToString***

11  
12           [C#]    public    virtual    GridColumnStylesCollection    GridColumnStyles    {get;}

13           [C++]    public:    \_\_property    virtual    GridColumnStylesCollection\*  
14           get\_GridColumnStyles();

15           [VB]    Overridable    Public    ReadOnly    Property    GridColumnStyles    As  
16           GridColumnStylesCollection

17           [JScript] public function get GridColumnStyles() : GridColumnStylesCollection;

18  
19           *Description*

20           Gets the collection of columns drawn for this table.

21           The **System.Windows.Forms.GridColumnStylesCollection** returned by the  
22           **System.Windows.Forms.DataGridTableStyle.GridColumnStyles** property  
23           allows you to create a customized set of column styles. For each  
24           **System.Data.DataColumn** in a **System.Data.DataTable** , set the  
25           **System.Windows.Forms.DataGridTableStyle.MappingName** of a  
         **System.Windows.Forms.DataGridColumnStyle** object to the  
         **System.Data.DataColumn.ColumnName** . That column style will

1 automatically be used when this  
2 **System.Windows.Forms.DataGridTableStyle** is displayed.

3 z) *GridLineColor*

4 aa) *ToString*

5 [C#] public Color GridLineColor {get; set;}

6 [C++] public: \_\_property Color get\_GridLineColor();public: \_\_property void  
7 set\_GridLineColor(Color);

8 [VB] Public Property GridLineColor As Color

9 [JScript] public function get GridLineColor() : Color;public function set  
10 GridLineColor(Color);

11  
12 *Description*

13 Gets or sets the color of grid lines.

14 bb) *GridLineStyle*

15 cc) *ToString*

16  
17 [C#] public DataGridLineStyle GridLineStyle {get; set;}

18 [C++] public: \_\_property DataGridLineStyle get\_GridLineStyle();public:  
19 \_\_property void set\_GridLineStyle(DataGridLineStyle);

20 [VB] Public Property GridLineStyle As DataGridLineStyle

21 [JScript] public function get GridLineStyle() : DataGridLineStyle;public function  
22 set  
23 GridLineStyle(DataGridLineStyle);

24  
25 *Description*



Gets or sets the style of grid lines.

*dd) HeaderBackColor*

*ee) ToString*

[C#] public Color HeaderBackColor {get; set;}

[C++] public: \_\_property Color get\_HeaderBackColor();public: \_\_property void set\_HeaderBackColor(Color);

[VB] Public Property HeaderBackColor As Color

[JScript] public function get HeaderBackColor() : Color;public function set HeaderBackColor(Color);

#### *Description*

Gets or sets the background color of headers.

*ff) HeaderFont*

*gg) ToString*

[C#] public Font HeaderFont {get; set;}

[C++] public: \_\_property Font\* get\_HeaderFont();public: \_\_property void set\_HeaderFont(Font\*);

[VB] Public Property HeaderFont As Font

[JScript] public function get HeaderFont() : Font;public function set HeaderFont(Font);

#### *Description*

Gets or sets the font used for header captions.

To set header caption text, use the **System.Windows.Forms.DataGridColumnStyle.HeaderText** property of the **System.Windows.Forms.DataGridColumnStyle** class.

*hh) HeaderForeColor*

*ii) ToString*

[C#]        public        Color        HeaderForeColor        {get;        set;}

[C++] public: \_\_property Color get\_HeaderForeColor();public: \_\_property void  
set\_HeaderForeColor(Color);

[VB]        Public        Property        HeaderForeColor        As        Color

[JScript] public function get HeaderForeColor() : Color;public function set  
HeaderForeColor(Color);

#### *Description*

Gets or sets the foreground color of headers.

*jj) LinkColor*

*kk) ToString*

[C#]        public        Color        LinkColor        {get;        set;}

[C++] public: \_\_property Color get\_LinkColor();public: \_\_property void  
set\_LinkColor(Color);

[VB]        Public        Property        LinkColor        As        Color

[JScript] public function get LinkColor() : Color;public function set  
LinkColor(Color);

*Description*

Gets or sets the color of link text.

*ll) LinkHoverColor*

*mm) ToString*

[C#]        public        Color        LinkHoverColor        {get;        set;}

[C++] public: \_\_property Color get\_LinkHoverColor();public: \_\_property void  
set\_LinkHoverColor(Color);

[VB]        Public        Property        LinkHoverColor        As        Color

[JScript] public function get LinkHoverColor() : Color;public function set  
LinkHoverColor(Color);

*Description*

Gets or sets the color displayed when hovering over link text.

*nn) MappingName*

*oo) ToString*

[C#]        public        string        MappingName        {get;        set;}

[C++] public: \_\_property String\* get\_MappingName();public: \_\_property void  
set\_MappingName(String\*);

[VB]        Public        Property        MappingName        As        String

[JScript] public function get MappingName() : String;public function set  
MappingName(String);

1  
2 *Description*

3 Gets or sets the name used to map this table to a specific data source.

4 The default is the name of the list managed by the  
5 **System.Windows.Forms.CurrencyManager** for this grid. The  
6 **System.Windows.Forms.CurrencyManager** for the  
7 **System.Windows.Forms.DataGridTableStyle** is set using the  
8 **System.Windows.Forms.DataGridTableStyle.#ctor** constructor.

9  
10 *pp) PreferredColumnWidth*

11 *qq) ToString*

12  
13 [C#] public int PreferredColumnWidth {get; set;}

14 [C++] public: \_\_property int get\_PreferredColumnWidth();public: \_\_property  
15 void set\_PreferredColumnWidth(int);

16 [VB] Public Property PreferredColumnWidth As Integer

17 [JScript] public function get PreferredColumnWidth() : int;public function set  
18 PreferredColumnWidth(int);

19  
20 *Description*

21 Gets or sets the width used to create columns when a new grid is displayed.

22 *rr) PreferredRowHeight*

23 *ss) ToString*

24  
25 [C#] public int PreferredRowHeight {get; set;}

[C++] public: \_\_property int get\_PreferredRowHeight();public: \_\_property void  
set\_PreferredRowHeight(int);

```

1 [VB]      Public      Property      PreferredRowHeight      As      Integer
2 [JScript] public function get PreferredRowHeight() : int;public function set
3 PreferredRowHeight(int);

```

#### *Description*

Gets or sets the height used to create a row when a new grid is displayed.

The preferred height is the minimum height needed to accommodate the displayed text with the assigned

**System.Windows.Forms.DataGridTableStyle.HeaderFont .**

*tt)   ReadOnly*

*uu)   ToString*

```

12 [C#]      public      virtual      bool      ReadOnly      {get;      set;}

```

```

13 [C++] public: __property virtual bool get_ReadOnly();public: __property virtual
14 void set_ReadOnly(bool);

```

```

15 [VB]      Overridable      Public      Property      ReadOnly      As      Boolean

```

```

16 [JScript] public function get ReadOnly() : Boolean;public function set
17 ReadOnly(Boolean);

```

#### *Description*

Gets or sets a value indicating whether columns can be edited.

You can also specify whether individual columns within the table are editable by setting the **System.Windows.Forms.DataGridColumnStyle** class's **System.Windows.Forms.DataGridColumnStyle.ReadOnly** property to an appropriate value, **true** or **false** .

1       vv)    *RowHeadersVisible*

2       ww)    *ToString*

3  
4   [C#]       public       bool       RowHeadersVisible       {get;       set;}

5   [C++] public: \_\_property bool get\_RowHeadersVisible();public: \_\_property void  
6   set\_RowHeadersVisible(bool);

7   [VB]       Public       Property       RowHeadersVisible       As       Boolean

8   [JScript] public function get RowHeadersVisible() : Boolean;public function set  
9   RowHeadersVisible(Boolean);

10  
11   *Description*

12   Gets or sets a value indicating whether row headers are visible.

13   When row headers are visible, a plus sign is displayed in each row header if the  
14   underlying **System.Data.DataTable** has a related child table.

15       xx)    *RowHeaderWidth*

16       yy)    *ToString*

17   [C#]       public       int       RowHeaderWidth       {get;       set;}

18   [C++] public: \_\_property int get\_RowHeaderWidth();public: \_\_property void  
19   set\_RowHeaderWidth(int);

20   [VB]       Public       Property       RowHeaderWidth       As       Integer

21   [JScript] public function get RowHeaderWidth() : int;public function set  
22   RowHeaderWidth(int);

23  
24   *Description*

25   Gets or sets the width of row headers.

When row headers are visible a plus sign is displayed in each row header if the underlying data has a related child table.

**zz) SelectionBackColor**

**aaa) ToString**

[C#]        public        Color        SelectionBackColor        {get;        set;}

[C++] public: \_\_property Color get\_SelectionBackColor();public: \_\_property void  
set\_SelectionBackColor(Color);

[VB]        Public        Property        SelectionBackColor        As        Color

[JScript] public function get SelectionBackColor() : Color;public function set  
SelectionBackColor(Color);

#### *Description*

Gets or sets the background color of selected cells.

**bbb) SelectionForeColor**

**ccc) ToString**

[C#]        public        Color        SelectionForeColor        {get;        set;}

[C++] public: \_\_property Color get\_SelectionForeColor();public: \_\_property void  
set\_SelectionForeColor(Color);

[VB]        Public        Property        SelectionForeColor        As        Color

[JScript] public function get SelectionForeColor() : Color;public function set  
SelectionForeColor(Color);

#### *Description*

Gets or sets the foreground color of selected cells.

*ddd) Site*

*eee) ToString*

#### *Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.AllowSorting** property value changes.

For more information about handling events, see .

*fff) ToString*

[C#]      public      event      EventHandler      AlternatingBackColorChanged;

[C++]      public:      \_\_event      EventHandler\*      AlternatingBackColorChanged;

[VB]      Public      Event      AlternatingBackColorChanged      As      EventHandler

#### *Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.AlternatingBackColor** value changes.

For more information about handling events, see .

*ggg) ToString*

[C#]      public      event      EventHandler      BackColorChanged;

[C++]      public:      \_\_event      EventHandler\*      BackColorChanged;

[VB]      Public      Event      BackColorChanged      As      EventHandler



1  
2 *Description*

3 Occurs when the **System.Windows.Forms.DataGridTableStyle.BackColor**  
4 value changes.

5 For more information about handling events, see .

6 *hhh) ToString*

7 [C#] public event EventHandler ColumnHeadersVisibleChanged;

8 [C++] public: \_\_event EventHandler\* ColumnHeadersVisibleChanged;

9 [VB] Public Event ColumnHeadersVisibleChanged As EventHandler

10  
11 *Description*

12 Occurs when the  
13 **System.Windows.Forms.DataGridTableStyle.ColumnHeadersVisible**  
14 value changes.

15 To set the caption text for each column in a grid, set the  
16 **System.Windows.Forms.DataGridColumnStyle.HeaderText** property of  
17 the **System.Windows.Forms.DataGridColumnStyle** class.

18  
19 *iii) ToString*

20 *Description*

21 Occurs when the **System.Windows.Forms.DataGridTableStyle.ForeColor**  
22 value changes.

23 For more information about handling events, see .  
24  
25

1            *jjj) ToString*

2  
3 [C#]       public       event       EventHandler       GridLineColorChanged;  
4 [C++]      public:      \_\_event      EventHandler\*      GridLineColorChanged;  
5 [VB]      Public      Event      GridLineColorChanged      As      EventHandler

6  
7            *Description*

8 Occurs when the  
9 **System.Windows.Forms.DataGridTableStyle.GridLineColor** value  
10 changes.

11 For more information about handling events, see .

12            *kkk) ToString*

13 [C#]       public       event       EventHandler       GridLineStyleChanged;  
14 [C++]      public:      \_\_event      EventHandler\*      GridLineStyleChanged;  
15 [VB]      Public      Event      GridLineStyleChanged      As      EventHandler

16  
17            *Description*

18 Occurs when the  
19 **System.Windows.Forms.DataGridTableStyle.GridLineStyle** value  
20 changes.

21 For more information about handling events, see .

22            *lll) ToString*

23 [C#]       public       event       EventHandler       HeaderBackColorChanged;  
24 [C++]      public:      \_\_event      EventHandler\*      HeaderBackColorChanged;  
25 [VB]      Public      Event      HeaderBackColorChanged      As      EventHandler

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.HeaderBackColor** value changes.

*mmm) ToString*

|       |         |         |                   |                    |
|-------|---------|---------|-------------------|--------------------|
| [C#]  | public  | event   | EventHandler      | HeaderFontChanged; |
| [C++] | public: | __event | EventHandler*     | HeaderFontChanged; |
| [VB]  | Public  | Event   | HeaderFontChanged | As EventHandler    |

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.HeaderFont** value changes.

For more information about handling events, see .

*nnn) ToString*

|       |         |         |                        |                         |
|-------|---------|---------|------------------------|-------------------------|
| [C#]  | public  | event   | EventHandler           | HeaderForeColorChanged; |
| [C++] | public: | __event | EventHandler*          | HeaderForeColorChanged; |
| [VB]  | Public  | Event   | HeaderForeColorChanged | As EventHandler         |

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.HeaderForeColor** value changes.

For more information about handling events, see .

ooo) ToString

```
[C#]      public      event      EventHandler      LinkColorChanged;  
[C++]     public:     __event     EventHandler*      LinkColorChanged;  
[VB]      Public      Event      LinkColorChanged     As      EventHandler
```

Description

Occurs when the **System.Windows.Forms.DataGridTableStyle.LinkColor** value changes.

For more information about handling events, see .

ppp) ToString

```
[C#]      public      event      EventHandler      LinkHoverColorChanged;  
[C++]     public:     __event     EventHandler*      LinkHoverColorChanged;  
[VB]      Public      Event      LinkHoverColorChanged     As      EventHandler
```

Description

Occurs when the **System.Windows.Forms.DataGridTableStyle.LinkHoverColor** value changes.

For more information about handling events, see .

qqq) ToString

```
[C#]      public      event      EventHandler      MappingNameChanged;  
[C++]     public:     __event     EventHandler*      MappingNameChanged;  
[VB]      Public      Event      MappingNameChanged     As      EventHandler
```

1  
2 *Description*

3 Occurs when the  
4 **System.Windows.Forms.DataGridTableStyle.MappingName** value  
5 changes.

6 For more information about handling events, see .

7  
8 *rrr) ToString*

9 [C#] public event EventHandler PreferredColumnWidthChanged;  
10 [C++] public: \_\_event EventHandler\* PreferredColumnWidthChanged;  
11 [VB] Public Event PreferredColumnWidthChanged As EventHandler

12 *Description*

13 Occurs when the  
14 **System.Windows.Forms.DataGridTableStyle.PreferredColumnWidth**  
15 property value changes.

16  
17 *sss) ToString*

18 [C#] public event EventHandler PreferredRowHeightChanged;  
19 [C++] public: \_\_event EventHandler\* PreferredRowHeightChanged;  
20 [VB] Public Event PreferredRowHeightChanged As EventHandler

21 *Description*

22 Occurs when the  
23 **System.Windows.Forms.DataGridTableStyle.PreferredRowHeight** value  
24 changes.

25 For more information about handling events, see .

1            *ttt) ToString*

2            [C#]        public        event        EventHandler        ReadOnlyChanged;  
3            [C++]        public:        \_\_event        EventHandler\*        ReadOnlyChanged;  
4            [VB]        Public        Event        ReadOnlyChanged        As        EventHandler

6            *Description*

7            Occurs when the **System.Windows.Forms.DataGridTableStyle.ReadOnly**  
8            value changes.

9            For more information about handling events, see .

10           *uuu) ToString*

12           [C#]        public        event        EventHandler        RowHeadersVisibleChanged;  
13           [C++]        public:        \_\_event        EventHandler\*        RowHeadersVisibleChanged;  
14           [VB]        Public        Event        RowHeadersVisibleChanged        As        EventHandler

16           *Description*

17           Occurs when the  
18           **System.Windows.Forms.DataGridTableStyle.RowHeadersVisible** value  
18           changes.

19           For more information about handling events, see .

20           *vvv) ToString*

22           [C#]        public        event        EventHandler        RowHeaderWidthChanged;  
23           [C++]        public:        \_\_event        EventHandler\*        RowHeaderWidthChanged;  
24           [VB]        Public        Event        RowHeaderWidthChanged        As        EventHandler

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.RowHeaderWidth** value changes.

For more information about handling events, see .

*www) ToString*

```
[C#]      public      event      EventHandler      SelectionBackColorChanged;  
[C++]     public:     __event   EventHandler*      SelectionBackColorChanged;  
[VB]      Public      Event      SelectionBackColorChanged      As      EventHandler
```

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.SelectionBackColor** value changes.

For more information about handling events, see .

*xxx) ToString*

```
[C#]      public      event      EventHandler      SelectionForeColorChanged;  
[C++]     public:     __event   EventHandler*      SelectionForeColorChanged;  
[VB]      Public      Event      SelectionForeColorChanged      As      EventHandler
```

*Description*

Occurs when the **System.Windows.Forms.DataGridTableStyle.SelectionForeColor** value changes.

### yyy) *BeginEdit*

```
1
2
3 [C#] public bool BeginEdit(DataGridColumnStyle gridColumn, int rowNumber);
4
5 [C++] public: __sealed bool BeginEdit(DataGridColumnStyle* gridColumn, int
6 rowNumber);
7
8 [VB] NotOverridable Public Function BeginEdit(ByVal gridColumn As
9 DataGridColumnStyle, ByVal rowNumber As Integer) As Boolean
10
11 [JScript] public function BeginEdit(gridColumn : DataGridColumnStyle,
12 rowNumber : int) : Boolean;
```

#### *Description*

Requests an edit operation.

*Return Value:* **true** , if the operation succeeds; otherwise, **false** .

The

**System.Windows.Forms.DataGridTableStyle.BeginEdit(System.Windows.Forms.DataGridColumnStyle, System.Int32)** method is intended to notify the **System.Windows.Forms.DataGrid** control when the user has begun an editing operation. When the controls is in edit mode, multiple edits can be made and the constraints will be temporarily unenforced. The **System.Windows.Forms.DataGridColumnStyle** to edit. The number of the edited row.

### zzz) *CreateGridColumn*

```
18
19
20 [C#] protected internal virtual DataGridColumnStyle
21 CreateGridColumn(PropertyDescriptor prop);
22
23 [C++] protected public: virtual DataGridColumnStyle*
24 CreateGridColumn(PropertyDescriptor* prop);
25
26 [VB] Overridable Protected Friend Dim Function CreateGridColumn(ByVal prop
27 As PropertyDescriptor) As DataGridColumnStyle
```



[JScript] package function CreateGridColumn(prop : PropertyDescriptor) :  
 DataGridColumnStyle; Creates a  
 System.Windows.Forms.DataGridColumnStyle .

#### Description

Creates a **System.Windows.Forms.DataGridColumnStyle** . using the specified property descriptor.

*Return Value:* The newly created **System.Windows.Forms.DataGridColumnStyle** object. The **System.ComponentModel.PropertyDescriptor** used to create the column style object.

#### aaaa) CreateGridColumn

[C#] protected internal virtual DataGridColumnStyle  
 CreateGridColumn(PropertyDescriptor prop, bool isDefault);  
 [C++] protected public: virtual DataGridColumnStyle\*  
 CreateGridColumn(PropertyDescriptor\* prop, bool isDefault);  
 [VB] Overridable Protected Friend Dim Function CreateGridColumn(ByVal prop  
 As PropertyDescriptor, ByVal isDefault As Boolean) As DataGridColumnStyle  
 [JScript] package function CreateGridColumn(prop : PropertyDescriptor,  
 isDefault : Boolean) : DataGridColumnStyle;

#### Description

Creates a **System.Windows.Forms.DataGridColumnStyle** using the specified property descriptor. Specifies whether the **System.Windows.Forms.DataGridColumnStyle** is a default column style.

*Return Value:* The newly created **System.Windows.Forms.DataGridColumnStyle** object. The **System.ComponentModel.PropertyDescriptor** used to create the column style object. Specifies whether the

**System.Windows.Forms.DataGridColumnStyle** object is a default column style. This parameter is read-only.

*bbbb) Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]      protected:      void      Dispose(bool      disposing);
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)
[JScript]      protected      override      function      Dispose(disposing      :      Boolean);
```

*Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.DataGridTableStyle**.

The method calls the **System.Windows.Forms.DataGridTableStyle.Dispose(System.Boolean)** method on each **System.Windows.Forms.DataGridColumnStyle** contained by the collection before calling **System.Windows.Forms.DataGrid.Dispose(System.Boolean)** on the grid itself.

*cccc) EndEdit*

```
[C#]      public      bool      EndEdit(DataGridColumnStyle      gridColumn,      int      rowNumber,
bool                                                                                                  shouldAbort);
[C++]      public:      __sealed      bool      EndEdit(DataGridColumnStyle*      gridColumn,      int
rowNumber,                                                                                              bool                                                                                                  shouldAbort);
[VB]      NotOverridable      Public      Function      EndEdit(ByVal      gridColumn      As
DataGridColumnStyle,      ByVal      rowNumber      As      Integer,      ByVal      shouldAbort      As
Boolean)                                                                                                  As      Boolean
[JScript]      public      function      EndEdit(gridColumn      :      DataGridColumnStyle,
```

rowNumber : int, shouldAbort : Boolean) : Boolean;

### *Description*

Requests an end to an edit operation.

*Return Value:* **true** if the edit operation ends successfully; otherwise, **false** .

Similar to the

**System.Windows.Forms.DataGridTableStyle.BeginEdit(System.Windows.Forms.DataGridColumnStyle,System.Int32)** method, the **System.Windows.Forms.DataGridTableStyle.EndEdit(System.Windows.Forms.DataGridColumnStyle,System.Int32,System.Boolean)** method is intended to notify the **System.Windows.Forms.DataGrid** when an edit operation is ending. The **System.Windows.Forms.DataGridColumnStyle** to edit. The number of the edited row. A value indicating whether the operation should be stopped; **true** if it should stop; otherwise, **false**.

### *dddd) OnAllowSortingChanged*

[C#] protected virtual void OnAllowSortingChanged(EventArgs e);

[C++] protected: virtual void OnAllowSortingChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnAllowSortingChanged(ByVal e As EventArgs)

[JScript] protected function OnAllowSortingChanged(e : EventArgs);

### *Description*

Raises the

**System.Windows.Forms.DataGridTableStyle.AllowSortingChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*eeee) OnColumnHeadersVisibleChanged*

[C#] protected virtual void OnColumnHeadersVisibleChanged(EventArgs e);  
[C++] protected: virtual void OnColumnHeadersVisibleChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnColumnHeadersVisibleChanged(ByVal e As EventArgs)  
[JScript] protected function OnColumnHeadersVisibleChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.DataGridTableStyle.ColumnHeadersVisibleChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*ffff) OnGridLineColorChanged*

[C#] protected virtual void OnGridLineColorChanged(EventArgs e);  
[C++] protected: virtual void OnGridLineColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnGridLineColorChanged(ByVal e As EventArgs)  
[JScript] protected function OnGridLineColorChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.DataGridTableStyle.GridLineColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*gggg) OnGridLineStyleChanged*

[C#] protected virtual void OnGridLineStyleChanged(EventArgs e);  
[C++] protected: virtual void OnGridLineStyleChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnGridLineStyleChanged(ByVal e As  
EventArgs)  
[JScript] protected function OnGridLineStyleChanged(e : EventArgs);

*Description*

Raises the  
**System.Windows.Forms.DataGridTableStyle.GridLineStyleChanged**  
event.

Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

*hhhh) OnHeaderBackColorChanged*

[C#] protected virtual void OnHeaderBackColorChanged(EventArgs e);  
[C++] protected: virtual void OnHeaderBackColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnHeaderBackColorChanged(ByVal e As  
EventArgs)  
[JScript] protected function OnHeaderBackColorChanged(e : EventArgs);

*Description*

Raises the  
**System.Windows.Forms.DataGridTableStyle.HeaderBackColorChanged**  
event.

Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

1                    **iii) OnHeaderFontChanged**

2 [C#]    protected    virtual    void    OnHeaderFontChanged(EventArgs e);  
3  
4 [C++]    protected:    virtual    void    OnHeaderFontChanged(EventArgs\* e);  
5  
6 [VB]    Overridable Protected Sub OnHeaderFontChanged(ByVal e As EventArgs)  
7  
8 [JScript]    protected    function    OnHeaderFontChanged(e : EventArgs);

9                    *Description*

10                  Raises the  
11                  **System.Windows.Forms.DataGridTableStyle.HeaderFontChanged** event.

12                  Raising an event invokes the event handler through a delegate. For more  
13                  information, see . An **System.EventArgs** that contains the event data.

14                    **iiii) OnHeaderForeColorChanged**

15 [C#]    protected    virtual    void    OnHeaderForeColorChanged(EventArgs e);  
16  
17 [C++]    protected:    virtual    void    OnHeaderForeColorChanged(EventArgs\* e);  
18  
19 [VB]    Overridable Protected Sub OnHeaderForeColorChanged(ByVal e As  
20    EventArgs)  
21  
22 [JScript]    protected    function    OnHeaderForeColorChanged(e : EventArgs);

23                    *Description*

24                  Raises the  
25                  **System.Windows.Forms.DataGridTableStyle.HeaderForeColorChanged**  
                  event.

                  Raising an event invokes the event handler through a delegate. For more  
                  information, see . An **System.EventArgs** that contains the event data.

#### *kkkk) OnLinkColorChanged*

[C#] protected virtual void OnLinkColorChanged(EventArgs e);  
[C++] protected: virtual void OnLinkColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnLinkColorChanged(ByVal e As EventArgs)  
[JScript] protected function OnLinkColorChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DataGridTableStyle.LinkColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *llll) OnLinkHoverColorChanged*

[C#] protected virtual void OnLinkHoverColorChanged(EventArgs e);  
[C++] protected: virtual void OnLinkHoverColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnLinkHoverColorChanged(ByVal e As EventArgs)  
[JScript] protected function OnLinkHoverColorChanged(e : EventArgs);

#### *Description*

Raises the LinkHoverColorChanged event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *mmmm)OnMappingNameChanged*

```
1  
2  
3 [C#] protected virtual void OnMappingNameChanged(EventArgs e);  
4 [C++] protected: virtual void OnMappingNameChanged(EventArgs* e);  
5 [VB] Overridable Protected Sub OnMappingNameChanged(ByVal e As  
6 EventArgs)  
7 [JScript] protected function OnMappingNameChanged(e : EventArgs);
```

#### *Description*

Raises the  
**System.Windows.Forms.DataGridTableStyle.MappingNameChanged**  
event

Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.

### *nnnn) OnPreferredColumnWidthChanged*

```
14  
15 [C#] protected virtual void OnPreferredColumnWidthChanged(EventArgs e);  
16 [C++] protected: virtual void OnPreferredColumnWidthChanged(EventArgs* e);  
17 [VB] Overridable Protected Sub OnPreferredColumnWidthChanged(ByVal e As  
18 EventArgs)  
19 [JScript] protected function OnPreferredColumnWidthChanged(e : EventArgs);  
20
```

#### *Description*

Raises the  
**System.Windows.Forms.DataGridTableStyle.PreferredColumnWidthCha  
nged** event.

Raising an event invokes the event handler through a delegate. For more  
information, see . An **System.EventArgs** that contains the event data.



### oooo) OnPreferredRowHeightChanged

[C#] protected virtual void OnPreferredRowHeightChanged(EventArgs e);  
[C++] protected: virtual void OnPreferredRowHeightChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnPreferredRowHeightChanged(ByVal e As EventArgs)  
[JScript] protected function OnPreferredRowHeightChanged(e : EventArgs);

#### Description

Raises the **System.Windows.Forms.DataGridTableStyle.PreferredRowHeightChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### pppp) OnReadOnlyChanged

[C#] protected virtual void OnReadOnlyChanged(EventArgs e);  
[C++] protected: virtual void OnReadOnlyChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnReadOnlyChanged(ByVal e As EventArgs)  
[JScript] protected function OnReadOnlyChanged(e : EventArgs);

#### Description

Raises the **System.Windows.Forms.DataGridTableStyle.ReadOnlyChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *qqqq) OnRowHeadersVisibleChanged*

[C#] protected virtual void OnRowHeadersVisibleChanged(EventArgs e);

[C++] protected: virtual void OnRowHeadersVisibleChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnRowHeadersVisibleChanged(ByVal e As EventArgs)

[JScript] protected function OnRowHeadersVisibleChanged(e : EventArgs);

#### *Description*

Raises the

**System.Windows.Forms.DataGridTableStyle.RowHeadersVisibleChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *rrrr) OnRowHeaderWidthChanged*

[C#] protected virtual void OnRowHeaderWidthChanged(EventArgs e);

[C++] protected: virtual void OnRowHeaderWidthChanged(EventArgs\* e);

[VB] Overridable Protected Sub OnRowHeaderWidthChanged(ByVal e As EventArgs)

[JScript] protected function OnRowHeaderWidthChanged(e : EventArgs);

#### *Description*

Raises the

**System.Windows.Forms.DataGridTableStyle.RowHeaderWidthChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*ssss) OnSelectionBackColorChanged*

[C#] protected virtual void OnSelectionBackColorChanged(EventArgs e);  
[C++] protected: virtual void OnSelectionBackColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnSelectionBackColorChanged(ByVal e As EventArgs)  
[JScript] protected function OnSelectionBackColorChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.DataGridTableStyle.SelectionBackColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*tttt) OnSelectionForeColorChanged*

[C#] protected virtual void OnSelectionForeColorChanged(EventArgs e);  
[C++] protected: virtual void OnSelectionForeColorChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnSelectionForeColorChanged(ByVal e As EventArgs)  
[JScript] protected function OnSelectionForeColorChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.DataGridTableStyle.SelectionForeColorChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

uuuu) *ResetAlternatingBackColor*

|           |         |          |                              |
|-----------|---------|----------|------------------------------|
| [C#]      | public  | void     | ResetAlternatingBackColor(); |
| [C++]     | public: | void     | ResetAlternatingBackColor(); |
| [VB]      | Public  | Sub      | ResetAlternatingBackColor()  |
| [JScript] | public  | function | ResetAlternatingBackColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.AlternatingBackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle**. You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeAlternatingBackColor** method to determine whether the property value has changed from its default.

vvvv) *ResetBackColor*

|           |         |          |                   |
|-----------|---------|----------|-------------------|
| [C#]      | public  | void     | ResetBackColor(); |
| [C++]     | public: | void     | ResetBackColor(); |
| [VB]      | Public  | Sub      | ResetBackColor()  |
| [JScript] | public  | function | ResetBackColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.BackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

*www)ResetForeColor*

|           |         |          |                   |
|-----------|---------|----------|-------------------|
| [C#]      | public  | void     | ResetForeColor(); |
| [C++]     | public: | void     | ResetForeColor(); |
| [VB]      | Public  | Sub      | ResetForeColor()  |
| [JScript] | public  | function | ResetForeColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.ForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** . You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeForeColor** method to determine whether the property value has changed from its default.

*xxxx) ResetGridLineColor*

|           |         |          |                       |
|-----------|---------|----------|-----------------------|
| [C#]      | public  | void     | ResetGridLineColor(); |
| [C++]     | public: | void     | ResetGridLineColor(); |
| [VB]      | Public  | Sub      | ResetGridLineColor()  |
| [JScript] | public  | function | ResetGridLineColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.GridLineColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** . You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeGridLineColor** or method to determine whether the property value has changed from its default.

*yyyy) ResetHeaderBackColor*

|           |         |          |                         |
|-----------|---------|----------|-------------------------|
| [C#]      | public  | void     | ResetHeaderBackColor(); |
| [C++]     | public: | void     | ResetHeaderBackColor(); |
| [VB]      | Public  | Sub      | ResetHeaderBackColor()  |
| [JScript] | public  | function | ResetHeaderBackColor(); |

#### *Description*

Resets the **System.Windows.Forms.DataGridTableStyle.HeaderBackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** . You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeHeaderBackColor** method to determine whether the property value has changed from its default.

*zzzz) ResetHeaderFont*

|           |         |          |                    |
|-----------|---------|----------|--------------------|
| [C#]      | public  | void     | ResetHeaderFont(); |
| [C++]     | public: | void     | ResetHeaderFont(); |
| [VB]      | Public  | Sub      | ResetHeaderFont()  |
| [JScript] | public  | function | ResetHeaderFont(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.HeaderFont** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

*aaaaa) ResetHeaderForeColor*

|           |         |          |                         |
|-----------|---------|----------|-------------------------|
| [C#]      | public  | void     | ResetHeaderForeColor(); |
| [C++]     | public: | void     | ResetHeaderForeColor(); |
| [VB]      | Public  | Sub      | ResetHeaderForeColor()  |
| [JScript] | public  | function | ResetHeaderForeColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.HeaderForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** . You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeHeaderForeColor** method to determine whether the property value has changed from its default.

*bbbbbb) ResetLinkColor*

|       |         |      |                   |
|-------|---------|------|-------------------|
| [C#]  | public  | void | ResetLinkColor(); |
| [C++] | public: | void | ResetLinkColor(); |
| [VB]  | Public  | Sub  | ResetLinkColor()  |

1 [JScript]                    public                    function                    ResetLinkColor();

3 *Description*

4 Resets the **System.Windows.Forms.DataGridTableStyle.LinkColor** property to its default value.

5 You typically use this method if you are either creating a designer for the  
6 **System.Windows.Forms.DataGridTableStyle** or creating your own control  
7 incorporating the **System.Windows.Forms.DataGridTableStyle** . You can  
8 use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeLinkColor**  
method to determine whether the property value has changed from its default.

9                    *cccc) ResetLinkHoverColor*

11 [C#]                    public                    void                    ResetLinkHoverColor();  
12 [C++]                    public:                    void                    ResetLinkHoverColor();  
13 [VB]                    Public                    Sub                    ResetLinkHoverColor()  
14 [JScript]                    public                    function                    ResetLinkHoverColor();

16 *Description*

17 Resets the **System.Windows.Forms.DataGridTableStyle.LinkHoverColor** property to its default value.

18 You typically use this method if you are either creating a designer for the  
19 **System.Windows.Forms.DataGridTableStyle** or creating your own control  
20 incorporating the **System.Windows.Forms.DataGridTableStyle** . You can  
21 use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeLinkHoverC**  
22 **olor** method to determine whether the property value has changed from its  
default.



*dddd) ResetSelectionBackColor*

|           |         |          |                            |
|-----------|---------|----------|----------------------------|
| [C#]      | public  | void     | ResetSelectionBackColor(); |
| [C++]     | public: | void     | ResetSelectionBackColor(); |
| [VB]      | Public  | Sub      | ResetSelectionBackColor()  |
| [JScript] | public  | function | ResetSelectionBackColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.SelectionBackColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle**. You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeSelectionBackColor** method to determine whether the property value has changed from its default.

*eeee) ResetSelectionForeColor*

|           |         |          |                            |
|-----------|---------|----------|----------------------------|
| [C#]      | public  | void     | ResetSelectionForeColor(); |
| [C++]     | public: | void     | ResetSelectionForeColor(); |
| [VB]      | Public  | Sub      | ResetSelectionForeColor()  |
| [JScript] | public  | function | ResetSelectionForeColor(); |

*Description*

Resets the **System.Windows.Forms.DataGridTableStyle.SelectionForeColor** property to its default value.

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** . You can use the **System.Windows.Forms.DataGridTableStyle.ShouldSerializeSelectionForeColor** method to determine whether the property value has changed from its default.

*fffff) ShouldSerializeAlternatingBackColor*

[C#]      protected      virtual      bool      ShouldSerializeAlternatingBackColor();  
 [C++]      protected:      virtual      bool      ShouldSerializeAlternatingBackColor();  
 [VB]      Overridable Protected Function ShouldSerializeAlternatingBackColor() As Boolean  
 [JScript]      protected      function      ShouldSerializeAlternatingBackColor() : Boolean;

#### *Description*

Indicates whether the **System.Windows.Forms.DataGridTableStyle.AlternatingBackColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

*ggggg) ShouldSerializeBackColor*

[C#]              protected              bool              ShouldSerializeBackColor();  
 [C++]              protected:              bool              ShouldSerializeBackColor();  
 [VB]      Protected      Function      ShouldSerializeBackColor()      As      Boolean  
 [JScript]      protected      function      ShouldSerializeBackColor()      :      Boolean;

## Description

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.BackColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the

**System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

### hhhhh)ShouldSerializeForeColor

|      |           |      |                             |
|------|-----------|------|-----------------------------|
| [C#] | protected | bool | ShouldSerializeForeColor(); |
|------|-----------|------|-----------------------------|

|       |            |      |                             |
|-------|------------|------|-----------------------------|
| [C++] | protected: | bool | ShouldSerializeForeColor(); |
|-------|------------|------|-----------------------------|

|      |           |          |                            |    |         |
|------|-----------|----------|----------------------------|----|---------|
| [VB] | Protected | Function | ShouldSerializeForeColor() | As | Boolean |
|------|-----------|----------|----------------------------|----|---------|

|           |           |          |                            |   |          |
|-----------|-----------|----------|----------------------------|---|----------|
| [JScript] | protected | function | ShouldSerializeForeColor() | : | Boolean; |
|-----------|-----------|----------|----------------------------|---|----------|

## Description

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.ForeColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the

**System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

### iiii) ShouldSerializeGridLineColor

|      |           |         |      |                                 |
|------|-----------|---------|------|---------------------------------|
| [C#] | protected | virtual | bool | ShouldSerializeGridLineColor(); |
|------|-----------|---------|------|---------------------------------|

|       |            |         |      |                                 |
|-------|------------|---------|------|---------------------------------|
| [C++] | protected: | virtual | bool | ShouldSerializeGridLineColor(); |
|-------|------------|---------|------|---------------------------------|

1 [VB] Overridable Protected Function ShouldSerializeGridLineColor() As Boolean

2 [JScript] protected function ShouldSerializeGridLineColor() : Boolean;

3  
4 *Description*

5 Indicates whether the

6 **System.Windows.Forms.DataGridTableStyle.GridLineColor** property  
should be persisted.

7 *Return Value:* **true** if the property value has changed from its default; otherwise,  
**false** .

8 You typically use this method if you are either creating a designer for the  
9 **System.Windows.Forms.DataGridTableStyle** or creating your own control  
incorporating the **System.Windows.Forms.DataGridTableStyle** .

10 *jjjj) ShouldSerializeHeaderBackColor*

11  
12 [C#] protected virtual bool ShouldSerializeHeaderBackColor();

13 [C++] protected: virtual bool ShouldSerializeHeaderBackColor();

14 [VB] Overridable Protected Function ShouldSerializeHeaderBackColor() As  
15 Boolean

16 [JScript] protected function ShouldSerializeHeaderBackColor() : Boolean;

17  
18 *Description*

19 Indicates whether the

20 **System.Windows.Forms.DataGridTableStyle.HeaderBackColor** property  
should be persisted.

21 *Return Value:* **true** if the property value has changed from its default; otherwise,  
**false** .

22 You typically use this method if you are either creating a designer for the  
23 **System.Windows.Forms.DataGridTableStyle** or creating your own control  
incorporating the **System.Windows.Forms.DataGridTableStyle** .  
24  
25

#### *kkkkk) ShouldSerializeHeaderForeColor*

[C#]      protected      virtual      bool      ShouldSerializeHeaderForeColor();  
[C++]      protected:      virtual      bool      ShouldSerializeHeaderForeColor();  
[VB]      Overridable      Protected      Function      ShouldSerializeHeaderForeColor()      As      Boolean  
[JScript]      protected      function      ShouldSerializeHeaderForeColor()      :      Boolean;

#### *Description*

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.HeaderForeColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

#### *lllll) ShouldSerializeLinkColor*

[C#]      protected      virtual      bool      ShouldSerializeLinkColor();  
[C++]      protected:      virtual      bool      ShouldSerializeLinkColor();  
[VB]      Overridable      Protected      Function      ShouldSerializeLinkColor()      As      Boolean  
[JScript]      protected      function      ShouldSerializeLinkColor()      :      Boolean;

#### *Description*

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.LinkColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

*mmmmm)ShouldSerializeLinkHoverColor*

[C#]       protected       virtual       bool       ShouldSerializeLinkHoverColor();

[C++]       protected:       virtual       bool       ShouldSerializeLinkHoverColor();

[VB]   Overridable   Protected   Function   ShouldSerializeLinkHoverColor()   As   Boolean

[JScript]   protected   function   ShouldSerializeLinkHoverColor()   :   Boolean;

#### *Description*

Indicates whether the **System.Windows.Forms.DataGridTableStyle.LinkHoverColor** property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the **System.Windows.Forms.DataGridTableStyle** or creating your own control incorporating the **System.Windows.Forms.DataGridTableStyle** .

*nnnnn)ShouldSerializePreferredRowHeight*

[C#]       protected       bool       ShouldSerializePreferredRowHeight();

[C++]       protected:       bool       ShouldSerializePreferredRowHeight();

[VB]   Protected   Function   ShouldSerializePreferredRowHeight()   As   Boolean

[JScript]   protected   function   ShouldSerializePreferredRowHeight()   :   Boolean;

#### *Description*

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.PreferredRowHeight**

property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the

**System.Windows.Forms.DataGridTableStyle** or creating your own control

incorporating the **System.Windows.Forms.DataGridTableStyle** .

*ooooo) ShouldSerializeSelectionBackColor*

[C#]           protected           bool           ShouldSerializeSelectionBackColor();

[C++]           protected:           bool           ShouldSerializeSelectionBackColor();

[VB] Protected Function ShouldSerializeSelectionBackColor() As Boolean

[JScript] protected function ShouldSerializeSelectionBackColor() : Boolean;

### *Description*

Indicates whether the

**System.Windows.Forms.DataGridTableStyle.SelectionBackColor**

property should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise, **false** .

You typically use this method if you are either creating a designer for the

**System.Windows.Forms.DataGridTableStyle** or creating your own control

incorporating the **System.Windows.Forms.DataGridTableStyle** .

*ppppp) ShouldSerializeSelectionForeColor*

[C#]           protected           virtual   bool           ShouldSerializeSelectionForeColor();

[C++]           protected:           virtual   bool           ShouldSerializeSelectionForeColor();

[VB] Overridable Protected Function ShouldSerializeSelectionForeColor() As

Boolean

1 [JScript] protected function ShouldSerializeSelectionForeColor() : Boolean;

2  
3 *Description*

4 Indicates whether the  
5 **System.Windows.Forms.DataGridTableStyle.SelectionForeColor** property  
6 should be persisted.

*Return Value:* **true** if the property value has changed from its default; otherwise,  
7 **false** .

8 You typically use this method if you are either creating a designer for the  
9 **System.Windows.Forms.DataGridTableStyle** or creating your own control  
10 incorporating the **System.Windows.Forms.DataGridTableStyle** .

11  
12  
13 DataGridTextBoxColumn class (System.Windows.Forms)

14 a) *ToString*

15  
16  
17 *Description*

18 Represents a **System.Windows.Forms.TextBox** control that is hosted in a  
19 **System.Windows.Forms.DataGridTextBoxColumn** .

20 The **System.Windows.Forms.DataGridTextBoxColumn** and the  
21 **System.Windows.Forms.DataGridTextBoxColumn** work together to allow  
22 users to directly edit values in a **System.Windows.Forms.DataGrid** control  
23 column. The **System.Windows.Forms.DataGridTextBoxColumn** derives  
24 from **System.Windows.Forms.DataGridColumnStyle** , and is designed to  
25 host the **System.Windows.Forms.DataGridTextBoxColumn** , which derives from the  
26 **System.Windows.Forms.TextBox** control.

27 b) *DataGridTextBoxColumn*

28 *Example Syntax:*

29 c) *ToString*

30  
31  
32 [C#] public DataGridTextBoxColumn();



|    |                                              |         |                    |
|----|----------------------------------------------|---------|--------------------|
| 1  | [C++]                                        | public: | DataGridTextBox(); |
| 2  | [VB]                                         | Public  | Sub New()          |
| 3  | [JScript] public function DataGridTextBox(); |         |                    |
| 4  |                                              |         |                    |
| 5  |                                              |         |                    |
| 6  |                                              |         |                    |
| 7  |                                              |         |                    |
| 8  |                                              |         |                    |
| 9  |                                              |         |                    |
| 10 |                                              |         |                    |
| 11 |                                              |         |                    |
| 12 |                                              |         |                    |
| 13 |                                              |         |                    |
| 14 |                                              |         |                    |
| 15 |                                              |         |                    |
| 16 |                                              |         |                    |
| 17 |                                              |         |                    |
| 18 |                                              |         |                    |
| 19 |                                              |         |                    |
| 20 |                                              |         |                    |
| 21 |                                              |         |                    |
| 22 |                                              |         |                    |
| 23 |                                              |         |                    |
| 24 |                                              |         |                    |
| 25 |                                              |         |                    |

- d) *AcceptsReturn*
- e) *AcceptsTab*
- f) *AccessibilityObject*
- g) *AccessibleDefaultActionDescription*
- h) *AccessibleDescription*
- i) *AccessibleName*
- j) *AccessibleRole*
- k) *AllowDrop*
- l) *Anchor*
- m) *AutoSize*
- n) *BackColor*
- o) *BackgroundImage*
- p) *BindingContext*
- q) *BorderStyle*
- r) *Bottom*
- s) *Bounds*
- t) *CanFocus*
- u) *CanSelect*
- v) *CanUndo*
- w) *Capture*
- x) *CausesValidation*
- y) *CharacterCasing*
- z) *ClientRectangle*

1        **aa)    ClientSize**

2        **bb)    CompanyName**

3        **cc)    Container**

4        **dd)    ContainsFocus**

5        **ee)    ContextMenu**

6        **ff)    Controls**

7        **gg)    Created**

8        **hh)    CreateParams**

9        **ii)    Cursor**

10       **jj)    DataBindings**

11       **kk)    DefaultImeMode**

12       **ll)    DefaultSize**

13       **mm)    DesignMode**

14       **nn)    DisplayRectangle**

15       **oo)    Disposing**

16       **pp)    Dock**

17       **qq)    Enabled**

18       **rr)    Events**

19       **ss)    Focused**

20       **tt)    Font**

21       **uu)    FontHeight**

22       **vv)    ForeColor**

23       **ww)    Handle**

xx) *HasChildren*

yy) *Height*

zz) *HideSelection*

aaa) *ImeMode*

bbb) *InvokeRequired*

ccc) *IsAccessible*

ddd) *IsDisposed*

eee) *IsHandleCreated*

fff) *IsInEditOrNavigateMode*

ggg) *ToString*

#### *Description*

Gets or sets a value indicating whether the **System.Windows.Forms.DataGridTextBox** is in a mode that allows either editing or navigating.

The **System.Windows.Forms.DataGridTextBox.IsInEditOrNavigateMode** property is used within the **System.Windows.Forms.Control.ProcessKeyMessage(System.Windows.Forms.Message@)** to determine how to process key press events, to check the state of the **System.Windows.Forms.DataGridTextBox** . For example, if one of the navigation (arrow) keys is pressed, the appropriate action for the state of the control must occur.

|    |                               |
|----|-------------------------------|
| 1  | <i>hhh) Left</i>              |
| 2  | <i>iii) Lines</i>             |
| 3  | <i>jjj) Location</i>          |
| 4  | <i>kkk) MaxLength</i>         |
| 5  | <i>lll) Modified</i>          |
| 6  | <i>mmm) Multiline</i>         |
| 7  | <i>nnn) Name</i>              |
| 8  | <i>ooo) Parent</i>            |
| 9  | <i>ppp) PasswordChar</i>      |
| 10 | <i>qqq) PreferredHeight</i>   |
| 11 | <i>rrr) ProductName</i>       |
| 12 | <i>sss) ProductVersion</i>    |
| 13 | <i>ttt) ReadOnly</i>          |
| 14 | <i>uuu) RecreatingHandle</i>  |
| 15 | <i>vvv) Region</i>            |
| 16 | <i>www) RenderRightToLeft</i> |
| 17 | <i>xxx) ResizeRedraw</i>      |
| 18 | <i>yyy) Right</i>             |
| 19 | <i>zzz) RightToLeft</i>       |
| 20 | <i>aaaa) ScrollBars</i>       |
| 21 | <i>bbbb) SelectedText</i>     |
| 22 | <i>cccc) SelectionLength</i>  |
| 23 | <i>dddd) SelectionStart</i>   |
| 24 |                               |
| 25 |                               |

1        *eeee) ShowFocusCues*

2        *ffff) ShowKeyboardCues*

3        *gggg) Site*

4        *hhhh) Size*

5        *iiii) TabIndex*

6        *jjjj) TabStop*

7        *kkkk) Tag*

8        *llll) Text*

9        *mmmm)TextAlign*

10       *nnnn) TextLength*

11       *oooo) Top*

12       *pppp) TopLevelControl*

13       *qqqq) Visible*

14       *rrrr) Width*

15       *ssss) WindowTarget*

16       *tttt) WordWrap*

17       *uuuu) OnKeyPress*

18  
19  
20    [C#]    protected    override    void    OnKeyPress(KeyPressEventArgs    e);

21    [C++]    protected:    void    OnKeyPress(KeyPressEventArgs\*    e);

22    [VB]    Overrides Protected Sub OnKeyPress(ByVal e As KeyPressEventArgs)

23    [JScript]    protected    override    function    OnKeyPress(e : KeyPressEventArgs);

## Description

Raises the **System.Windows.Forms.Control.KeyPress** event. A **System.Windows.Forms.KeyPressEventArgs** that contains the event data.

### *vvvv) OnMouseWheel*

[C#] protected override void OnMouseWheel(MouseEventArgs e);  
[C++] protected: void OnMouseWheel(MouseEventArgs\* e);  
[VB] Overrides Protected Sub OnMouseWheel(ByVal e As MouseEventArgs)  
[JScript] protected override function OnMouseWheel(e : MouseEventArgs);

## Description

Raises the **System.Windows.Forms.Control.MouseWheel** event. A **System.Windows.Forms.MouseEventArgs** that contains the event data.

### *www)ProcessKeyMessage*

[C#] protected internal override bool ProcessKeyMessage(ref Message m);  
[C++] protected public: bool ProcessKeyMessage(Message\* m);  
[VB] Overrides Protected Friend Dim Function ProcessKeyMessage(ByRef m As Message)  
As Boolean  
[JScript] package override function ProcessKeyMessage(m : Message) : Boolean;

## Description

Indicates whether the key pressed is processed further (for example, to navigate, or escape). This property is read-only.

**Return Value:** **true** if the key is consumed, **false** to if the key is further processed.

This method is called when a control receives a keyboard message. A **System.Windows.Forms.Message**, passed by reference, that contains the key data.

*xxxx) SetDataGrid*

[C#] public void SetDataGrid(DataGrid parentGrid);

[C++] public: void SetDataGrid(DataGrid\* parentGrid);

[VB] Public Sub SetDataGrid(ByVal parentGrid As DataGrid)

[JScript] public function SetDataGrid(parentGrid : DataGrid);

*Description*

Sets the **System.Windows.Forms.DataGrid** to which this **System.Windows.Forms.TextBox** control belongs. The **System.Windows.Forms.DataGrid** control that hosts the control.

*yyyy) WndProc*

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message\* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

*Description*

Raises the **System.Windows.Forms.Control.WndProc(System.Windows.Forms.Message@)** event. A **System.Windows.Forms.Message** that contains the event data.



DataGridTextBoxColumn class (System.Windows.Forms)

a) *WndProc*

*Description*

Hosts a **System.Windows.Forms.TextBox** control in a cell of a **System.Windows.Forms.DataGridColumnStyle** for editing strings.

The **System.Windows.Forms.DataGridTextBoxColumn** class derives from the **abstract** class **System.Windows.Forms.DataGridColumnStyle**. At run time, the **System.Windows.Forms.DataGridTextBoxColumn** hosts a **System.Windows.Forms.DataGridTextBoxColumn** control that allows users to edit text.

b) *DataGridTextBoxColumn*

*Example Syntax:*

c) *WndProc*

[C#]                      public                      DataGridTextBoxColumn();

[C++]                    public:                    DataGridTextBoxColumn();

[VB]                    Public                    Sub                    New()

[JScript] public function DataGridTextBoxColumn(); Initializes a new instance of the **System.Windows.Forms.DataGridTextBoxColumn** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.DataGridTextBoxColumn** class.

d) *DataGridTextBoxColumn*

*Example Syntax:*

e) *WndProc*

```
[C#]      public      DataGridTextBoxColumn(PropertyDescriptor      prop);  
[C++]     public:     DataGridTextBoxColumn(PropertyDescriptor*      prop);  
[VB]      Public      Sub      New(ByVal      prop      As      PropertyDescriptor)  
[JScript] public function DataGridTextBoxColumn(prop : PropertyDescriptor);
```

*Description*

Initializes a new instance of a **System.Windows.Forms.DataGridTextBoxColumn** with a specified **System.ComponentModel.PropertyDescriptor**.

The **System.Windows.Forms.DataGridColumnStyle** uses a **System.ComponentModel.PropertyDescriptor** to determine the type of data displayed in the column. To return a **System.ComponentModel.PropertyDescriptorCollection**, use the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** class. The **System.ComponentModel.PropertyDescriptor** for the column with which the **System.Windows.Forms.DataGridTextBoxColumn** will be associated.

f) *DataGridTextBoxColumn*

*Example Syntax:*

g) *WndProc*

```
[C#] public DataGridTextBoxColumn(PropertyDescriptor prop, bool isDefault);  
[C++] public: DataGridTextBoxColumn(PropertyDescriptor* prop, bool  
isDefault);  
[VB] Public Sub New(ByVal prop As PropertyDescriptor, ByVal isDefault As  
Boolean)  
[JScript] public function DataGridTextBoxColumn(prop : PropertyDescriptor,
```

isDefault : Boolean);

### *Description*

Initializes a new instance of the **System.Windows.Forms.DataGridTextBoxColumn** class using the specified **System.ComponentModel.PropertyDescriptor** . Specifies whether the **System.Windows.Forms.DataGridTextBoxColumn** is a default column.

The **System.Windows.Forms.DataGridColumnStyle** uses a **System.ComponentModel.PropertyDescriptor** to determine the type of data displayed in the column. To return a **System.ComponentModel.PropertyDescriptorCollection** , use the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** class. The **System.ComponentModel.PropertyDescriptor** to be associated with the **System.Windows.Forms.DataGridTextBoxColumn**. Specifies whether the **System.Windows.Forms.DataGridTextBoxColumn** is a default column.

#### *h) DataGridTextBoxColumn*

#### *Example Syntax:*

#### *i) WndProc*

[C#] public DataGridTextBoxColumn(PropertyDescriptor prop, string format);

[C++] public: DataGridTextBoxColumn(PropertyDescriptor\* prop, String\* format);

[VB] Public Sub New(ByVal prop As PropertyDescriptor, ByVal format As String)

[JScript] public function DataGridTextBoxColumn(prop : PropertyDescriptor, format : String);

### *Description*

Initializes a new instance of a

**System.Windows.Forms.DataGridTextBoxColumn** with the specified **System.ComponentModel.PropertyDescriptor** and format.

Use this constructor to create a custom format for the displayed data. The **System.ComponentModel.PropertyDescriptor** for the column with which the **System.Windows.Forms.DataGridTextBoxColumn** will be associated. The format used to format the column values.

j) *DataGridTextBoxColumn*

*Example Syntax:*

k) *WndProc*

```
[C#] public DataGridTextBoxColumn(PropertyDescriptor prop, string format,
bool isDefault);
```

```
[C++] public: DataGridTextBoxColumn(PropertyDescriptor* prop, String*
format, bool isDefault);
```

```
[VB] Public Sub New(ByVal prop As PropertyDescriptor, ByVal format As
String, ByVal isDefault As Boolean)
```

```
[JScript] public function DataGridTextBoxColumn(prop : PropertyDescriptor,
format : String, isDefault : Boolean);
```

### *Description*

Initializes a new instance of the **System.Windows.Forms.DataGridTextBoxColumn** class with a specified **System.ComponentModel.PropertyDescriptor** and format. Specifies whether the column is the default column.

The **System.Windows.Forms.DataGridColumnStyle** uses a **System.ComponentModel.PropertyDescriptor** to determine the type of data displayed in the column. To return a **System.ComponentModel.PropertyDescriptorCollection**, use the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** class. The

**System.ComponentModel.PropertyDescriptor** to be associated with the **System.Windows.Forms.DataGridTextBoxColumn**. The format used. Specifies whether the **System.Windows.Forms.DataGridTextBoxColumn** is the default column.

- l) Alignment*
- m) Container*
- n) DataGridTableStyle*
- o) DesignMode*
- p) Events*
- q) FontHeight*
- r) Format*
- s) WndProc*

#### *Description*

Gets or sets the character(s) that specify how text is formatted.

The **System.Windows.Forms.DataGridTextBoxColumn.Format** property specifies how values are displayed in the column. For example, set the property to "c" to specify that the values will be formatted as the local currency. The **System.Globalization.CultureInfo** for the computer is used to determine the actual currency format. The values are automatically unformatted to the native type when data is changed.

- t) FormatInfo*
- u) WndProc*

```
[C#]      public      IFormatProvider      FormatInfo      {get;      set;}
[C++] public: __property IFormatProvider* get_FormatInfo();public: __property
void      set_FormatInfo(IFormatProvider*);
```

[VB]      Public      Property      FormatInfo      As      IFormatProvider  
[JScript] public function get FormatInfo() : IFormatProvider; public function set  
FormatInfo(IFormatProvider);

#### *Description*

Gets or sets the culture specific information used to determine how values are formatted.

When setting the

**System.Windows.Forms.DataGridTextBoxColumn.Format** property to one of the formatting characters, the

**System.Windows.Forms.DataGridTextBoxColumn** uses the information provided by the

**System.Windows.Forms.DataGridTextBoxColumn.FormatInfo** property to further specify what cultural-specific formatting to use. For example, when the

**System.Windows.Forms.DataGridTextBoxColumn.Format** property is set to the format character "c" (for currency), you can further specify that the symbol for the lira be used. To do this, create a new

**System.Globalization.CultureInfo** object with the locale ID for Italy, and set the **System.Windows.Forms.DataGridTextBoxColumn.FormatInfo** property to the new **System.Globalization.CultureInfo** object.

v)      *HeaderAccessibleObject*

w)      *HeaderText*

x)      *MappingName*

y)      *NullText*

z)      *PropertyDescriptor*

aa)     *WndProc*

#### *Description*

Gets or sets the **System.ComponentModel.PropertyDescriptor** for the **System.Windows.Forms.DataGridTextBoxColumn**.

**cc) *WndProc***

```
[C++] public: __property virtual bool get_ReadOnly();public: __property virtual  
void set_ReadOnly(bool);
```

```
[JScript] public function get ReadOnly() : Boolean;public function set
ReadOnly(Boolean);
```

Sets a value indicating whether the text box column is read-only.

*ee) TextBox*

***ff) WndProc***

Gets the hosted **System.Windows.Forms.TextBox** control.

### ***hh) Abort***

[VB] Overrides Protected Friend Dim Sub Abort(ByVal rowNum As Integer)

1 [JScript] package override function Abort(rowNum : int);

2  
3 *Description*

4 Initiates a request to interrupt an edit procedure.

5 This method rolls back any change made to the column and invokes the  
6 **System.Windows.Forms.DataGridTextBoxColumn.HideEditBox** and  
7 **System.Windows.Forms.DataGridTextBoxColumn.EndEdit** methods. The  
8 number of the row in which an edit operation is being interrupted.

9  
10 *ii) Commit*

11 [C#] protected internal override bool Commit(CurrencyManager dataSource, int  
12 rowNum);

13 [C++] protected public: bool Commit(CurrencyManager\* dataSource, int  
14 rowNum);

15 [VB] Overrides Protected Friend Dim Function Commit(ByVal dataSource As  
16 CurrencyManager, ByVal rowNum As Integer) As Boolean

17 [JScript] package override function Commit(dataSource : CurrencyManager,  
18 rowNum : int) : Boolean;

19 *Description*

20 Initiates a request to complete an editing procedure.

21 *Return Value:* **true** if the value was successfully committed; otherwise, **false** .

22 The method checks to ensure that an edit is indeed occurring. If so, it formats  
23 the value appropriately. If the value is **null** , the method enters  
24 **System.Convert.DBNull** into the column; otherwise, it uses the  
25 **System.Windows.Forms.DataGridColumnStyle.SetColumnValueAtRow(  
System.Windows.Forms.CurrencyManager, System.Int32, System.Object  
)** method to commit the value. The  
**System.Windows.Forms.CurrencyManager** of the



**System.Windows.Forms.DataGrid** control the column belongs to. The number of the edited row.

*jj) ConcedeFocus*

|           |           |           |          |                 |                 |
|-----------|-----------|-----------|----------|-----------------|-----------------|
| [C#]      | protected | internal  | override | void            | ConcedeFocus(); |
| [C++]     | protected | public:   | void     | ConcedeFocus(); |                 |
| [VB]      | Overrides | Protected | Friend   | Dim Sub         | ConcedeFocus()  |
| [JScript] | package   | override  | function | ConcedeFocus(); |                 |

*Description*

Informs the column that the focus is being conceded.

In this overridden method, the **System.Windows.Forms.TextBox** control hosted by the **System.Windows.Forms.DataGridTextBoxColumn** is hidden.

*kk) Edit*

|           |           |           |          |                                                                                                                                     |                                                                                                                                                                                  |
|-----------|-----------|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [C#]      | protected | internal  | override | void                                                                                                                                | Edit(CurrencyManager source, int rowNum, Rectangle bounds, bool readOnly, string instantText, bool cellsVisible);                                                                |
| [C++]     | protected | public:   | void     | Edit(CurrencyManager* source, int rowNum, Rectangle bounds, bool readOnly, String* instantText, bool cellsVisible);                 |                                                                                                                                                                                  |
| [VB]      | Overrides | Protected | Friend   | Dim Sub                                                                                                                             | Edit(ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal bounds As Rectangle, ByVal readOnly As Boolean, ByVal instantText As String, ByVal cellsVisible As Boolean) |
| [JScript] | package   | override  | function | Edit(source : CurrencyManager, rowNum : int, bounds : Rectangle, readOnly : Boolean, instantText : String, cellsVisible : Boolean); |                                                                                                                                                                                  |

## Description

Prepares a cell for editing.

The

**System.Windows.Forms.DataGridColumnStyle.Edit(System.Windows.Forms.CurrencyManager, System.Int32, System.Drawing.Rectangle, System.Boolean)** sites a **System.Windows.Forms.TextBox** control on the grid at the location of the cell being edited. The method is called by the **System.Windows.Forms.DataGrid** class's **System.Windows.Forms.DataGrid.BeginEdit(System.Windows.Forms.DataGridColumnStyle, System.Int32)** method when an editing operation is about to begin. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** control the column belongs to. The row number in this column being edited. The bounding **System.Drawing.Rectangle** in which the control is to be sited. A value indicating whether the column is a read-only. The text to display in the control. A value indicating whether the cell is visible.

## II) EndEdit

|           |            |          |            |
|-----------|------------|----------|------------|
| [C#]      | protected  | void     | EndEdit(); |
| [C++]     | protected: | void     | EndEdit(); |
| [VB]      | Protected  | Sub      | EndEdit()  |
| [JScript] | protected  | function | EndEdit(); |

## Description

Ends an edit operation on the **System.Windows.Forms.DataGridColumnStyle**.

To edit the value of a cell, call the **System.Data.DataRow** object's **System.Data.DataRow.BeginEdit** before changing the value. You must invoke the **System.Data.DataRow.AcceptChanges** method on both the **System.Data.DataRow** and **System.Data.DataTable** objects before the change is committed.

*mm) EnterNullValue*

```
[C#]      protected      internal      override      void      EnterNullValue();
[C++]      protected      public:      void      EnterNullValue();
[VB]      Overrides      Protected      Friend      Dim      Sub      EnterNullValue()
[JScript]      package      override      function      EnterNullValue();
```

*Description*

Enters a **System.DBNull.Value** in the column.

Enters the **System.Windows.Forms.DataGridColumnStyle.NullText** value into the cell.

*nn) GetMinimumHeight*

```
[C#]      protected      internal      override      int      GetMinimumHeight();
[C++]      protected      public:      int      GetMinimumHeight();
[VB]      Overrides      Protected      Friend      Dim      Function      GetMinimumHeight() As Integer
[JScript]      package      override      function      GetMinimumHeight()      :      int;
```

*Description*

Gets the height of a cell in a **System.Windows.Forms.DataGridColumnStyle**.

*Return Value:* The height of a cell.

The value returned by

**System.Windows.Forms.DataGridTextBoxColumn.GetMinimumHeight** is calculated by adding the size of the column's font plus a margin value.

oo) *GetPreferredHeight*

[C#] protected internal override int GetPreferredHeight(Graphics g, object value);  
[C++] protected public: int GetPreferredHeight(Graphics\* g, Object\* value);  
[VB] Overrides Protected Friend Dim Function GetPreferredHeight(ByVal g As Graphics, ByVal value As Object) As Integer  
[JScript] package override function GetPreferredHeight(g : Graphics, value : Object) : int;

*Description*

Gets the height to be used in for automatically resizing columns.

*Return Value:* The height the cells automatically resize to. A

**System.Drawing.Graphics** object used to draw shapes on the screen. The value to draw.

pp) *GetPreferredSize*

[C#] protected internal override Size GetPreferredSize(Graphics g, object value);  
[C++] protected public: Size GetPreferredSize(Graphics\* g, Object\* value);  
[VB] Overrides Protected Friend Dim Function GetPreferredSize(ByVal g As Graphics, ByVal value As Object) As Size  
[JScript] package override function GetPreferredSize(g : Graphics, value : Object) : Size;

*Description*

Returns the optimum width and height of the cell in a specified row relative to the specified value.

*Return Value:* A **System.Drawing.Size** that contains the dimensions of the cell.

The optimum width and height is calculated by measuring the string size, taking into account its font and attributes, and adding margin values. A

**System.Drawing.Graphics** object used to draw shapes on the screen. The value to draw.

*qq) HideEditBox*

|           |            |          |                |
|-----------|------------|----------|----------------|
| [C#]      | protected  | void     | HideEditBox(); |
| [C++]     | protected: | void     | HideEditBox(); |
| [VB]      | Protected  | Sub      | HideEditBox()  |
| [JScript] | protected  | function | HideEditBox(); |

*Description*

Hides the **System.Windows.Forms.DataGridTextBox** control and moves the focus to the **System.Windows.Forms.DataGrid** control.

*rr) Paint*

|           |                                                                                                                                                                                   |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [C#]      | protected internal override void Paint(Graphics g, Rectangle bounds, CurrencyManager source, int rowNum);                                                                         |
| [C++]     | protected public: void Paint(Graphics* g, Rectangle bounds, CurrencyManager* source, int rowNum);                                                                                 |
| [VB]      | Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As Integer)                                |
| [JScript] | package override function Paint(g : Graphics, bounds : Rectangle, source : CurrencyManager, rowNum : int); Paints the column in the <b>System.Windows.Forms.DataGrid</b> control. |

## Description

Paints the a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics** , **System.Drawing.Rectangle** , **System.Windows.Forms.CurrencyManager** , and row number. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** the that contains the column. The number of the row in the underlying data table.

## ss) Paint

[C#] protected internal override void Paint(Graphics g, Rectangle bounds, CurrencyManager source, int rowNum, bool alignToRight);

[C++] protected public: void Paint(Graphics\* g, Rectangle bounds, CurrencyManager\* source, int rowNum, bool alignToRight);

[VB] Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal alignToRight As Boolean)

[JScript] package override function Paint(g : Graphics, bounds : Rectangle, source : CurrencyManager, rowNum : int, alignToRight : Boolean);

## Description

Paints a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics** , **System.Drawing.Rectangle** , **System.Windows.Forms.CurrencyManager** , row number, and alignment.

The **System.Windows.Forms.DataGridTextBoxColumn.Paint(System.Drawing.Graphics, System.Drawing.Rectangle, System.Windows.Forms.CurrencyManager, System.Int32)** method uses the **System.Windows.Forms.DataGridColumnStyle.GetColumnValueAtRow**

**System.Windows.Forms.CurrencyManager, System.Int32)** to determine the value to draw in the cell. The **System.Windows.Forms.DataGridTextBoxColumn.PaintText(System.Drawing.Graphics, System.Drawing.Rectangle, System.String, System.Boolean)** method is called to draw the cell and its contents. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** that contains the column. The number of the row in the underlying data table. A value indicating whether to align the column's content to the right.

*tt) Paint*

[C#] protected internal override void Paint(Graphics g, Rectangle bounds, CurrencyManager source, int rowNum, Brush backBrush, Brush foreBrush, bool alignToRight);

[C++] protected public: void Paint(Graphics\* g, Rectangle bounds, CurrencyManager\* source, int rowNum, Brush\* backBrush, Brush\* foreBrush, bool alignToRight);

[VB] Overrides Protected Friend Dim Sub Paint(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal backBrush As Brush, ByVal foreBrush As Brush, ByVal alignToRight As Boolean)

[JScript] package override function Paint(g : Graphics, bounds : Rectangle, source : CurrencyManager, rowNum : int, backBrush : Brush, foreBrush : Brush, alignToRight : Boolean);

*Description*

Paints a **System.Windows.Forms.DataGridColumnStyle** with the specified **System.Drawing.Graphics** , **System.Drawing.Rectangle** ,

**System.Windows.Forms.CurrencyManager** , row number,  
**System.Drawing.Brush** , and foreground color.

The **System.Windows.Forms.DataGridTextBoxColumn.Paint(System.Drawing.Graphics, System.Drawing.Rectangle, System.Windows.Forms.CurrencyManager, System.Int32)** method uses the **System.Windows.Forms.DataGridColumnStyle.GetColumnValueAtRow(System.Windows.Forms.CurrencyManager, System.Int32)** to determine the value to draw in the cell. The **System.Windows.Forms.DataGridTextBoxColumn.PaintText(System.Drawing.Graphics, System.Drawing.Rectangle, System.String, System.Boolean)** method is called to draw the cell and its contents. The **System.Drawing.Graphics** object to draw to. The bounding **System.Drawing.Rectangle** to paint into. The **System.Windows.Forms.CurrencyManager** of the **System.Windows.Forms.DataGrid** that contains the column. The number of the row in the underlying data table. A **System.Drawing.Brush** that paints the background. A **System.Drawing.Brush** that paints the foreground color. A value indicating whether to align the column's content to the right.

*uu) PaintText*

[C#] protected void PaintText(Graphics g, Rectangle bounds, string text, bool alignToRight);

[C++] protected: void PaintText(Graphics\* g, Rectangle bounds, String\* text, bool alignToRight);

[VB] Protected Sub PaintText(ByVal g As Graphics, ByVal bounds As Rectangle, ByVal text As String, ByVal alignToRight As Boolean)

[JScript] protected function PaintText(g : Graphics, bounds : Rectangle, text : String, alignToRight : Boolean); Draws the specified text and surrounding rectangle at the specified location.

*Description*

Draws the text and rectangle at the given location with the specified alignment.



The

**System.Windows.Forms.DataGridTextBoxColumn.PaintText(System.Drawing.Graphics, System.Drawing.Rectangle, System.String, System.Boolean)** method uses the **System.Windows.Forms.DataFormats.Format** object set with the **System.Windows.Forms.DataGridTextBoxColumn.Format** property to format the value before drawing it to the screen. A **System.Drawing.Graphics** object used to draw the string. A **System.Drawing.Rectangle** which contains the boundary data of the rectangle. The string to be drawn to the screen. A value indicating whether the text is right-aligned.

vv) *PaintText*

[C#] protected void PaintText(Graphics g, Rectangle textBounds, string text,

Brush backBrush, Brush foreBrush, bool alignToRight);

[C++] protected: void PaintText(Graphics\* g, Rectangle textBounds, String\* text,

Brush\* backBrush, Brush\* foreBrush, bool alignToRight);

[VB] Protected Sub PaintText(ByVal g As Graphics, ByVal textBounds As Rectangle, ByVal text As String, ByVal backBrush As Brush, ByVal foreBrush As Brush, ByVal alignToRight As Boolean)

[JScript] protected function PaintText(g : Graphics, textBounds : Rectangle, text : String, backBrush : Brush, foreBrush : Brush, alignToRight : Boolean);

*Description*

Draws the text and rectangle at the specified location with the specified colors and alignment.

The

**System.Windows.Forms.DataGridTextBoxColumn.PaintText(System.Drawing.Graphics, System.Drawing.Rectangle, System.String, System.Boolean)** method uses the **System.Windows.Forms.DataGridTextBoxColumn.Format** property to format the value before drawing it to the screen. A **System.Drawing.Graphics** object used to draw the string. A **System.Drawing.Rectangle** which contains the boundary data of the rectangle. The string to be drawn to the screen. A

**System.Drawing.Brush** that determines the rectangle's background color A **System.Drawing.Brush** that determines the rectangles foreground color. A value indicating whether the text is right-aligned.

*ww) SetDataGridInColumn*

[C#] protected override void SetDataGridInColumn(DataGrid value);

[C++] protected: void SetDataGridInColumn(DataGrid\* value);

[VB] Overrides Protected Sub SetDataGridInColumn(ByVal value As DataGrid)

[JScript] protected override function SetDataGridInColumn(value : DataGrid);

#### *Description*

Adds a **System.Windows.Forms.TextBox** control to the **System.Windows.Forms.DataGrid** control's **System.Windows.Forms.Control.ControlCollection** .

When this methods is called, the hosted **System.Windows.Forms.TextBox** control is added to the **System.Windows.Forms.DataGrid** control's **System.Windows.Forms.Control.ControlCollection** . This allows the **System.Windows.Forms.CurrencyManager** to associate both controls with the same data source. The **System.Windows.Forms.DataGrid** control the **System.Windows.Forms.TextBox** control is added to.

*xx) UpdateUI*

[C#] protected internal override void UpdateUI(CurrencyManager source, int rowNum, string instantText);

[C++] protected public: void UpdateUI(CurrencyManager\* source, int rowNum, String\* instantText);

[VB] Overrides Protected Friend Dim Sub UpdateUI(ByVal source As CurrencyManager, ByVal rowNum As Integer, ByVal instantText As String)

[JScript] package override function UpdateUI(source : CurrencyManager,

rowNum : int, instantText : String);

### Description

Updates the user interface. The **System.Windows.Forms.CurrencyManager** that supplies the data. The index of the row to update. The text that will be displayed in the cell.

DataObject class (System.Windows.Forms)

#### a) UpdateUI

### Description

Implements a basic data transfer mechanism.

**System.Windows.Forms.DataObject** implements the **System.Windows.Forms.IDataObject** interface, whose methods provide a format-independent mechanism for data transfer.

#### b) DataObject

#### Example Syntax:

#### c) UpdateUI

[C#] public DataObject();

[C++] public: DataObject();

[VB] Public Sub New()

[JScript] public function DataObject(); Initializes a new instance of the **System.Windows.Forms.DataObject** class.

### Description

1 Initializes a new instance of the **System.Windows.Forms.DataObject** class,  
which can store arbitrary data.

2 *d) DataObject*

3 *Example Syntax:*

4 *e) UpdateUI*

6 [C#] public DataObject(object data);

7 [C++] public: DataObject(Object\* data);

8 [VB] Public Sub New(ByVal data As Object)

9 [JScript] public function DataObject(data : Object);

11 *Description*

12 Initializes a new instance of the **System.Windows.Forms.DataObject** class,  
13 containing the specified data. The data to store.

14 *f) DataObject*

15 *Example Syntax:*

16 *g) UpdateUI*

18 [C#] public DataObject(string format, object data);

19 [C++] public: DataObject(String\* format, Object\* data);

20 [VB] Public Sub New(ByVal format As String, ByVal data As Object)

21 [JScript] public function DataObject(format : String, data : Object);

23 *Description*

24 Initializes a new instance of the **System.Windows.Forms.DataObject** class,  
25 containing the specified data and its associated format. The class type associated

with the data. See **System.Windows.Forms.DataFormats** for the predefined formats. The data to store.

*h) GetData*

```
[C#]      public      virtual      object      GetData(string      format);  
[C++]     public:     virtual      Object*      GetData(String*      format);  
[VB] Overridable Public Function GetData(ByVal format As String) As Object  
[JScript] public      function      GetData(format      :      String)      :      Object;
```

*Description*

Returns the data associated with the specified data format.

*Return Value:* The data associated with the specified format, or **null** .

If this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if the data was stored with automatic conversion set to **false** , this method returns **null** . The format of the data to retrieve. See

**System.Windows.Forms.DataFormats** for predefined formats.

*i) GetData*

```
[C#]      public      virtual      object      GetData(Type      format);  
[C++]     public:     virtual      Object*      GetData(Type*      format);  
[VB] Overridable Public Function GetData(ByVal format As Type) As Object  
[JScript] public      function      GetData(format      :      Type)      :      Object;
```

*Description*

Returns the data associated with the specified class type format.

*Return Value:* The data associated with the specified format, or **null** .

If this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if

the data was stored with automatic conversion set to **false** , this method returns **null** . A **System.Type** representing the format of the data to retrieve.

j) *GetData*

[C#] public virtual object GetData(string format, bool autoConvert);

[C++] public: virtual Object\* GetData(String\* format, bool autoConvert);

[VB] Overridable Public Function GetData(ByVal format As String, ByVal autoConvert As Boolean) As Object

[JScript] public function GetData(format : String, autoConvert : Boolean) : Object;

Returns the data associated with the specified data format.

*Description*

Returns the data associated with the specified data format, using an automated conversion parameter to determine whether to convert the data to the format.

*Return Value:* The data associated with the specified format, or **null** .

If the *autoConvert* parameter is **true** and this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if the data was stored with the automatic conversion set to **false** , this method returns **null** . The format of the data to retrieve. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to the convert data to the specified format; otherwise, **false**.

k) *GetDataPresent*

[C#] public virtual bool GetDataPresent(string format);

[C++] public: virtual bool GetDataPresent(String\* format);

[VB] Overridable Public Function GetDataPresent(ByVal format As String) As Boolean

[JScript] public function GetDataPresent(format : String) : Boolean;

## Description

Determines whether data stored in this instance is associated with, or can be converted to, the specified format.

**Return Value:** **true** if data stored in this instance is associated with, or can be converted to, the specified format; otherwise, **false** .

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.DataObject.GetData(System.String,System.Boolean)** . Call **System.Windows.Forms.DataObject.GetFormats(System.Boolean)** for the formats that are available in this instance. The format to check for. See **System.Windows.Forms.DataFormats** for predefined formats.

### 1) *GetDataPresent*

[C#]      public      virtual      bool      GetDataPresent(Type      format);

[C++]      public:      virtual      bool      GetDataPresent(Type\*      format);

[VB] Overridable Public Function GetDataPresent(ByVal format As Type) As Boolean

[JScript] public function GetDataPresent(format : Type) : Boolean; Determines whether data stored in this instance is associated with the specified format.

## Description

Determines whether data stored in this instance is associated with, or can be converted to, the specified format.

**Return Value:** **true** if data stored in this instance is associated with, or can be converted to, the specified format; otherwise, **false** .

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.DataObject.GetData(System.String,System.Boolean)** . Call **System.Windows.Forms.DataObject.GetFormats(System.Boolean)** for

the formats that are available in this instance. A **System.Type** representing the format to check for.

*m) GetDataPresent*

[C#] public virtual bool GetDataPresent(string format, bool autoConvert);

[C++] public: virtual bool GetDataPresent(String\* format, bool autoConvert);

[VB] Overridable Public Function GetDataPresent(ByVal format As String,  
ByVal autoConvert As Boolean) As Boolean

[JScript] public function GetDataPresent(format : String, autoConvert : Boolean) :  
Boolean;

*Description*

Determines whether data stored in this instance is associated with the specified format, using an automatic conversion parameter to determine whether to convert the data to the format.

*Return Value:* **true** if the data is in, or can be converted to, the specified format; otherwise, **false**.

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.DataObject.GetData(System.String, System.Boolean)**. Call **System.Windows.Forms.DataObject.GetFormats(System.Boolean)** for the formats that are available in this instance. The format to check for. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to determine whether data stored in this instance can be converted to the specified format; **false** to check whether the data is in the specified format.

*n) GetFormats*

[C#] public virtual string[] GetFormats();

[C++] public: virtual String\* GetFormats() \_\_gc[];

[VB] Overridable Public Function GetFormats() As String()



1 [JScript] public function GetFormats() : String[];

3 *Description*

4 Returns a list of all formats that data stored in this instance is associated with or  
can be converted to.

5 *Return Value:* An array of the names representing a list of all formats that are  
supported by the data stored in this object.

6 Call this method to get the supported data formats before calling  
7 **System.Windows.Forms.DataObject.GetData(System.String, System.Boolean)** . See **System.Windows.Forms.DataFormats** for the predefined  
8 formats.

9 o) *GetFormats*

11 [C#] public virtual string[] GetFormats(bool autoConvert);

12 [C++] public: virtual String\* GetFormats(bool autoConvert) \_\_gc[];

13 [VB] Overridable Public Function GetFormats(ByVal autoConvert As Boolean)

14 As String()

15 [JScript] public function GetFormats(autoConvert : Boolean) : String[]; Returns a  
16 list of all formats that data stored in this instance is associated with or can be  
17 converted to.

19 *Description*

20 Returns a list of all formats that data stored in this instance is associated with or  
can be converted to, using an automatic conversion parameter to determine  
21 whether to retrieve only native data formats or all formats that the data can be  
converted to.

22 *Return Value:* An array of the names representing a list of all formats that are  
23 supported by the data stored in this object.

24 Call this method to get the supported data formats before calling  
25 **System.Windows.Forms.DataObject.GetData(System.String, System.Boolean)** . See **System.Windows.Forms.DataFormats** for the predefined

formats. **true** to retrieve all formats that data stored in this instance is associated with, or can be converted to; **false** to retrieve only native data formats.

*p) SetData*

[C#] public virtual void SetData(object data);

[C++] public: virtual void SetData(Object\* data);

[VB] Overridable Public Sub SetData(ByVal data As Object)

[JScript] public function SetData(data : Object);

*Description*

Stores the specified data in this instance, using the class of the data for the format.

The data format is its class. If you do not know the format of the target application, you can store data in multiple formats using this method. The data to store.

*q) SetData*

[C#] public virtual void SetData(string format, object data);

[C++] public: virtual void SetData(String\* format, Object\* data);

[VB] Overridable Public Sub SetData(ByVal format As String, ByVal data As Object)

[JScript] public function SetData(format : String, data : Object);

*Description*

Stores the specified format and data in this instance.

If you do not know the format of the target application, you can store data in multiple formats using this method. The format associated with the data. See

**System.Windows.Forms.DataFormats** for predefined formats. The data to store.

*r) SetData*

[C#] public virtual void SetData(Type format, object data);

[C++] public: virtual void SetData(Type\* format, Object\* data);

[VB] Overridable Public Sub SetData(ByVal format As Type, ByVal data As Object)

[JScript] public function SetData(format : Type, data : Object);

*Description*

Stores the specified data and its associated class type in this instance.

If you do not know the format of the target application, you can store data in multiple formats using this method. A **System.Type** representing the format associated with the data. The data to store.

*s) SetData*

[C#] public virtual void SetData(string format, bool autoConvert, object data);

[C++] public: virtual void SetData(String\* format, bool autoConvert, Object\* data);

[VB] Overridable Public Sub SetData(ByVal format As String, ByVal autoConvert As Boolean, ByVal data As Object)

[JScript] public function SetData(format : String, autoConvert : Boolean, data : Object); Stores the specified format and data in this instance.

*Description*

Stores the specified format and data in this instance, using the automatic conversion parameter to specify whether the data can be converted to another format.

If you do not know the format of the target application, you can store data in multiple formats using this method. The format associated with the data. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to allow the data to be converted to another format; otherwise, **false**. The data to store.

*t) UnsafeNativeMethods.IOleDataObject.OleDAdvise*

[C#] int

UnsafeNativeMethods.IOleDataObject.OleDAdvise(NativeMethods.FORMATETC pFormatetc, int advf, object pAdvSink, int[] pdwConnection);

[C++] int

UnsafeNativeMethods::IOleDataObject::OleDAdvise(NativeMethods.FORMATETC\* pFormatetc, int advf, Object\* pAdvSink, int pdwConnection \_\_gc[]);

[VB] Function IOleDataObject.OleDAdvise(ByVal pFormatetc As NativeMethods.FORMATETC, ByVal advf As Integer, ByVal pAdvSink As Object, ByVal pdwConnection() As Integer) As Integer Implements UnsafeNativeMethods.IOleDataObject.OleDAdvise

[JScript] function UnsafeNativeMethods.IOleDataObject.OleDAdvise(pFormatetc : NativeMethods.FORMATETC, advf : int, pAdvSink : Object, pdwConnection : int[]) : int;

*u) UnsafeNativeMethods.IOleDataObject.OleDUnadvise*

[C#] int UnsafeNativeMethods.IOleDataObject.OleDUnadvise(int dwConnection);

[C++] int UnsafeNativeMethods::IOleDataObject::OleDUnadvise(int

```

1 dwConnection);
2 [VB] Function IOleDataObject.OleDUnadvise(ByVal dwConnection As Integer)
3 As Integer Implements UnsafeNativeMethods.IOleDataObject.OleDUnadvise
4 [JScript] function
5 UnsafeNativeMethods.IOleDataObject.OleDUnadvise(dwConnection : int) : int;
6
7
8 v) UnsafeNativeMethods.IOleDataObject.OleEnumDAdvise
9
10 [C#] int UnsafeNativeMethods.IOleDataObject.OleEnumDAdvise(object[]
11 ppenumAdvise);
12
13 [C++] int UnsafeNativeMethods::IOleDataObject::OleEnumDAdvise(Object*
14 ppenumAdvise __gc[]);
15 [VB] Function IOleDataObject.OleEnumDAdvise(ByVal ppenumAdvise() As
16 Object) As Integer Implements
17 UnsafeNativeMethods.IOleDataObject.OleEnumDAdvise
18 [JScript] function
19 UnsafeNativeMethods.IOleDataObject.OleEnumDAdvise(ppenumAdvise :
20 Object[]) : int;
21
22 w) UnsafeNativeMethods.IOleDataObject.OleEnumFormatEtc
23
24 [C#] IEnumFORMATETC
25 UnsafeNativeMethods.IOleDataObject.OleEnumFormatEtc(int dwDirection);
26
27 [C++] IEnumFORMATETC*
28 UnsafeNativeMethods::IOleDataObject::OleEnumFormatEtc(int dwDirection);
29 [VB] Function IOleDataObject.OleEnumFormatEtc(ByVal dwDirection As

```

```

1 Integer)          As          IEnumFORMATETC          Implements
2 UnsafeNativeMethods.IOleDataObject.OleEnumFormatEtc
3 [JScript]                                                function
4 UnsafeNativeMethods.IOleDataObject.OleEnumFormatEtc(dwDirection : int) :
5 IEnumFORMATETC;
6
7
8 x)      UnsafeNativeMethods.IOleDataObject.OleGetCanonicalFormatEtc
9
10
11 [C#]                                                int
12 UnsafeNativeMethods.IOleDataObject.OleGetCanonicalFormatEtc(NativeMethod
13 s.FORMATETC pformatetcIn, NativeMethods.FORMATETC pformatetcOut);
14
15 [C++]                                                int
16 UnsafeNativeMethods::IOleDataObject::OleGetCanonicalFormatEtc(NativeMetho
17 ds.FORMATETC* pformatetcIn, NativeMethods.FORMATETC*
18 pformatetcOut);
19 [VB] Function IOleDataObject.OleGetCanonicalFormatEtc(ByVal pformatetcIn
20 As NativeMethods.FORMATETC, ByVal pformatetcOut As
21 NativeMethods.FORMATETC) As Integer Implements
22 UnsafeNativeMethods.IOleDataObject.OleGetCanonicalFormatEtc
23 [JScript]                                                function
24 UnsafeNativeMethods.IOleDataObject.OleGetCanonicalFormatEtc(pformatetcIn :
25 NativeMethods.FORMATETC, pformatetcOut : NativeMethods.FORMATETC) :
26 int;

```

y) *UnsafeNativeMethods.IOleDataObject.OleGetData*

```

[C#] int
UnsafeNativeMethods.IOleDataObject.OleGetData(NativeMethods.FORMATET
C formatetc, NativeMethods.STGMEDIUM medium);
[C++] int
UnsafeNativeMethods::IOleDataObject::OleGetData(NativeMethods.FORMATE
TC* formatetc, NativeMethods.STGMEDIUM* medium);
[VB] Function IOleDataObject.OleGetData(ByVal formatetc As
NativeMethods.FORMATETC, ByVal medium As
NativeMethods.STGMEDIUM) As Integer Implements
UnsafeNativeMethods.IOleDataObject.OleGetData
[JScript] function UnsafeNativeMethods.IOleDataObject.OleGetData(formatetc :
NativeMethods.FORMATETC, medium : NativeMethods.STGMEDIUM) : int;

```

z) *UnsafeNativeMethods.IOleDataObject.OleGetDataHere*

```

[C#] int
UnsafeNativeMethods.IOleDataObject.OleGetDataHere(NativeMethods.FORMA
TETC formatetc, NativeMethods.STGMEDIUM medium);
[C++] int
UnsafeNativeMethods::IOleDataObject::OleGetDataHere(NativeMethods.FORM
ATETC* formatetc, NativeMethods.STGMEDIUM* medium);
[VB] Function IOleDataObject.OleGetDataHere(ByVal formatetc As
NativeMethods.FORMATETC, ByVal medium As
NativeMethods.STGMEDIUM) As Integer Implements

```

```

1 UnsafeNativeMethods.IOleDataObject.OleGetDataHere
2 [JScript] function
3 UnsafeNativeMethods.IOleDataObject.OleGetDataHere(formatetc :
4 NativeMethods.FORMATETC, medium : NativeMethods.STGMEDIUM) : int;

```

**aa) UnsafeNativeMethods.IOleDataObject.OleQueryGetData**

```

7 [C#] int
8 UnsafeNativeMethods.IOleDataObject.OleQueryGetData(NativeMethods.FORM
9 ATETC formatetc);

```

```

10 [C++] int
11 UnsafeNativeMethods::IOleDataObject::OleQueryGetData(NativeMethods.FOR
12 MATETC* formatetc);

```

```

13 [VB] Function IOleDataObject.OleQueryGetData(ByVal formatetc As
14 NativeMethods.FORMATETC) As Integer Implements

```

```

15 UnsafeNativeMethods.IOleDataObject.OleQueryGetData
16 [JScript] function

```

```

17 UnsafeNativeMethods.IOleDataObject.OleQueryGetData(formatetc :
18 NativeMethods.FORMATETC) : int;

```

**bb) UnsafeNativeMethods.IOleDataObject.OleSetData**

```

21 [C#] int
22 UnsafeNativeMethods.IOleDataObject.OleSetData(NativeMethods.FORMATET
23 C pFormatetcIn, NativeMethods.STGMEDIUM pmedium, int fRelease);

```

```

24 [C++] int

```



```

1 UnsafeNativeMethods::IOleDataObject::OleSetData(NativeMethods.FORMATET
2 C* pFormatetcIn, NativeMethods.STGMEDIUM* pmedium, int fRelease);
3 [VB] Function IOleDataObject.OleSetData(ByVal pFormatetcIn As
4 NativeMethods.FORMATETC, ByVal pmedium As
5 NativeMethods.STGMEDIUM, ByVal fRelease As Integer) As Integer
6 Implements UnsafeNativeMethods.IOleDataObject.OleSetData
7 [JScript] function
8 UnsafeNativeMethods.IOleDataObject.OleSetData(pFormatetcIn :
9 NativeMethods.FORMATETC, pmedium : NativeMethods.STGMEDIUM,
10 fRelease : int) : int;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

```

DateBoldEventArgs class (System.Windows.Forms)

- a) *ToString*
- b) *DaysToBold*
- c) *ToString*
- d) *Size*
- e) *ToString*
- f) *StartDate*
- g) *ToString*

DateBoldEventHandler delegate (System.Windows.Forms)

- a) *ToString*

DateRangeEventArgs class (System.Windows.Forms)

- a) *ToString*

### *Description*

Provides data for the **System.Windows.Forms.MonthCalendar.DateChanged** or **System.Windows.Forms.MonthCalendar.DateSelected** events of the **System.Windows.Forms.MonthCalendar** control.

The **System.Windows.Forms.MonthCalendar.DateChanged** event occurs when the currently selected date or range of dates changes; for example, when the user explicitly changes a selection within the current month or when the selection is implicitly changed in response to next/previous month navigation. The **System.Windows.Forms.MonthCalendar.DateSelected** event occurs when the user explicitly changes a selection. The **System.Windows.Forms.DateRangeEventArgs.#ctor** constructor specifies the start and end for the new date range that has been selected.

b) *DateRangeEventArgs*

*Example Syntax:*

c) *ToString*

[C#] public DateRangeEventArgs(DateTime start, DateTime end);

[C++] public: DateRangeEventArgs(DateTime start, DateTime end);

[VB] Public Sub New(ByVal start As DateTime, ByVal end As DateTime)

[JScript] public function DateRangeEventArgs(start : DateTime, end : DateTime);

*Description*

Initializes a new instance of the **System.Windows.Forms.DateRangeEventArgs** class.

The **System.Windows.Forms.DateRangeEventArgs.Start** and **System.Windows.Forms.DateRangeEventArgs.End** property values are set by the *start* and *end* parameter values of this constructor. The first date-time value in the range that the user has selected. The last date-time value in the range that the user has selected.

d) *End*

e) *ToString*

[C#] public DateTime End {get;}

[C++] public: \_\_property DateTime get\_End();

[VB] Public ReadOnly Property End As DateTime

[JScript] public function get End() : DateTime;

*Description*

Gets the last date-time value in the range that the user has selected.

f) *Start*

g) *ToString*

|           |         |            |          |                     |
|-----------|---------|------------|----------|---------------------|
| [C#]      | public  | DateTime   | Start    | {get;}              |
| [C++]     | public: | __property | DateTime | get_Start();        |
| [VB]      | Public  | ReadOnly   | Property | Start As DateTime   |
| [JScript] | public  | function   | get      | Start() : DateTime; |

*Description*

Gets the first date-time value in the range that the user has selected.

DateTimeRangeEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **System.Windows.Forms.MonthCalendar.DateChanged** or **System.Windows.Forms.MonthCalendar.DateSelected** event of a **System.Windows.Forms.MonthCalendar** . The source of the event. A **System.Windows.Forms.DateRangeEventArgs** that contains the event data.

When you create a **System.Windows.Forms.DateRangeEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

DateTimePicker class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a Windows date-time picker control.

The **System.Windows.Forms.DateTimePicker** control is used to allow the user to select a date and time, and to display that date-time in the specified format. You can limit the date and times that can be selected by setting the **System.Windows.Forms.DateTimePicker.MinDate** and **System.Windows.Forms.DateTimePicker.MaxDate** properties.

*b) ToString*

[C#]     protected     static     readonly     Color     DefaultMonthBackColor;

[C++]     protected:     static     Color     DefaultMonthBackColor;

[VB]     Protected     Shared     ReadOnly     DefaultMonthBackColor     As     Color

[JScript]     protected     static     var     DefaultMonthBackColor     :     Color;

*Description*

Specifies the default month background color of the date-time picker control. This field is read-only.

*c) ToString*

[C#]     protected     static     readonly     Color     DefaultTitleBackColor;

[C++]     protected:     static     Color     DefaultTitleBackColor;

[VB]     Protected     Shared     ReadOnly     DefaultTitleBackColor     As     Color

[JScript]     protected     static     var     DefaultTitleBackColor     :     Color;

*Description*

Specifies the default title back color of the date-time picker control. This field is read-only.

*d) ToString*

```
[C#]    protected    static    readonly    Color    DefaultTitleForeColor;
[C++]    protected:    static    Color    DefaultTitleForeColor;
[VB]    Protected    Shared    ReadOnly    DefaultTitleForeColor    As    Color
[JScript]    protected    static    var    DefaultTitleForeColor    :    Color;
```

*Description*

Specifies the default title foreground color of the date-time picker control. This field is read-only.

*e) ToString*

```
[C#]    protected    static    readonly    Color    DefaultTrailingForeColor;
[C++]    protected:    static    Color    DefaultTrailingForeColor;
[VB]    Protected    Shared    ReadOnly    DefaultTrailingForeColor    As    Color
[JScript]    protected    static    var    DefaultTrailingForeColor    :    Color;
```

*Description*

Specifies the default trailing foreground color of the date-time picker control. This field is read-only.

**f) ToString**

```
[C#]      public      static      readonly      DateTime      MaxDateTime;
[C++]      public:      static      DateTime      MaxDateTime;
[VB]      Public      Shared      ReadOnly      MaxDateTime      As      DateTime
[JScript]      public      static      var      MaxDateTime      :      DateTime;
```

**Description**

Specifies the maximum date value of the date-time picker control. This field is read-only.

The maximum date is set to 12/31/9998 23:59:59.

**g) ToString**

```
[C#]      public      static      readonly      DateTime      MinDateTime;
[C++]      public:      static      DateTime      MinDateTime;
[VB]      Public      Shared      ReadOnly      MinDateTime      As      DateTime
[JScript]      public      static      var      MinDateTime      :      DateTime;
```

**Description**

Specifies the minimum date value of the date-time picker control. This field is read-only.

The minimum date is set to 1/1/1753 00:00:00.

**h) DateTimePicker**

*Example Syntax:*

i) *ToString*

|           |         |                            |
|-----------|---------|----------------------------|
| [C#]      | public  | DateTimePicker();          |
| [C++]     | public: | DateTimePicker();          |
| [VB]      | Public  | Sub New()                  |
| [JScript] | public  | function DateTimePicker(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.DateTimePicker** class.

j) *AccessibilityObject*

k) *AccessibleDefaultActionDescription*

l) *AccessibleDescription*

m) *AccessibleName*

n) *AccessibleRole*

o) *AllowDrop*

p) *Anchor*

q) *BackColor*

r) *ToString*

*Description*



s) *BackgroundImage*

t) *ToString*

[C#] public override Image BackgroundImage {get; set;}

[C++] public: \_\_property virtual Image\* get\_BackgroundImage();public:

\_\_property virtual void set\_BackgroundImage(Image\*);

[VB] Overrides Public Property BackgroundImage As Image

[JScript] public function get BackgroundImage() : Image;public function set

BackgroundImage(Image);

### *Description*

u) *BindingContext*

v) *Bottom*

w) *Bounds*

x) *CalendarFont*

y) *ToString*

### *Description*

Gets or sets the font style applied to the calendar.

z) *CalendarForeColor*

aa) *ToString*

[C#] public Color CalendarForeColor {get; set;}

[C++] public: \_\_property Color get\_CalendarForeColor();public: \_\_property void  
set\_CalendarForeColor(Color);

[VB] Public Property CalendarForeColor As Color

[JScript] public function get CalendarForeColor() : Color;public function set  
CalendarForeColor(Color);

*Description*

Gets or sets the foreground color of the calendar.

When a **System.Windows.Forms.DateTimePicker** is created, this property is initially set equal to the **System.Windows.Forms.Control.ForeColor** property value.

bb) *CalendarMonthBackground*

cc) *ToString*

[C#] public Color CalendarMonthBackground {get; set;}

[C++] public: \_\_property Color get\_CalendarMonthBackground();public:  
\_\_property void set\_CalendarMonthBackground(Color);

[VB] Public Property CalendarMonthBackground As Color

[JScript] public function get CalendarMonthBackground() : Color;public function  
set CalendarMonthBackground(Color);

*Description*

Gets or sets the background color of the calendar month.

When a **System.Windows.Forms.DateTimePicker** is created, this property is initially set equal to the **System.Windows.Forms.DateTimePicker.DefaultMonthBackColor** field value.

*dd) CalendarTitleBackColor*

*ee) ToString*

```
[C#]      public      Color      CalendarTitleBackColor      {get;      set;}
```

```
[C++] public: __property Color get_CalendarTitleBackColor();public: __property  
void set_CalendarTitleBackColor(Color);
```

```
[VB]      Public      Property      CalendarTitleBackColor      As      Color
```

```
[JScript] public function get CalendarTitleBackColor() : Color;public function set  
CalendarTitleBackColor(Color);
```

#### *Description*

Gets or sets the background color of the calendar title.

When a **System.Windows.Forms.DateTimePicker** is created, this property is initially set equal to the **System.Windows.Forms.DateTimePicker.DefaultTitleBackColor** field value.

*ff) CalendarTitleForeColor*

*gg) ToString*

```
[C#]      public      Color      CalendarTitleForeColor      {get;      set;}
```

```
[C++] public: __property Color get_CalendarTitleForeColor();public: __property  
void set_CalendarTitleForeColor(Color);
```

```

1 [VB]      Public      Property      CalendarTitleForeColor      As      Color
2 [JScript] public function get CalendarTitleForeColor() : Color;public function set
3 CalendarTitleForeColor(Color);

```

#### *Description*

Gets or sets the foreground color of the calendar title.

When a **System.Windows.Forms.DateTimePicker** is created, this property is initially set equal to the **System.Windows.Forms.DateTimePicker.DefaultTitleForeColor** field value.

*hh) CalendarTrailingForeColor*

*ii) ToString*

```

12
13 [C#]      public      Color      CalendarTrailingForeColor      {get;      set;}
14 [C++]      public:      __property      Color      get_CalendarTrailingForeColor();public:
15 __property      void      set_CalendarTrailingForeColor(Color);
16 [VB]      Public      Property      CalendarTrailingForeColor      As      Color
17 [JScript] public function get CalendarTrailingForeColor() : Color;public function
18 set      CalendarTrailingForeColor(Color);

```

#### *Description*

Gets or sets the foreground color of the calendar trailing dates.

When a **System.Windows.Forms.DateTimePicker** is created, this property is initially set equal to the **System.Windows.Forms.DateTimePicker.DefaultTrailingForeColor** field value.

1        *jj)      CanFocus*

2        *kk)      CanSelect*

3        *ll)      Capture*

4        *mm)     CausesValidation*

5        *nn)      Checked*

6        *oo)      ToString*

7  
8  
9        *Description*

10       Gets or sets a value indicating whether the  
11       **System.Windows.Forms.DateTimePicker.Value** property has been set with  
12       a valid date-time value and the displayed value is able to be updated.

13       This parameter is used to obtain the state of the check box that is displayed if  
14       the **System.Windows.Forms.DateTimePicker.ShowCheckBox** property  
15       value is **true** . If the **System.Windows.Forms.DateTimePicker.Checked**  
16       property value is **true** , the date-time picker control displays the properly  
17       formatted **System.Windows.Forms.DateTimePicker.Value** property value;  
18       otherwise, the control displays either the current date and time ( **System.DateTime.Now** ) or the last valid date-time value assigned to the  
19       **System.Windows.Forms.DateTimePicker.Value** property.  
20  
21  
22  
23  
24  
25

1        *pp) ClientRectangle*

2        *qq) ClientSize*

3        *rr) CompanyName*

4        *ss) Container*

5        *tt) ContainsFocus*

6        *uu) ContextMenu*

7        *vv) Controls*

8        *ww) Created*

9        *xx) CreateParams*

10       *yy) ToString*

11  
12  
13       *Description*

14       Returns the CreateParams used to create this window.

15  
16       *zz) Cursor*

17       *aaa) CustomFormat*

18       *bbb) ToString*

19  
20  
21       *Description*

22       Gets or sets the custom date-time format string.

23       To display string literals that contain date and time separators or format strings  
24       you must escape the substring. For example, to display the date as "June 06 at  
25       3:00 PM", set the

**System.Windows.Forms.DateTimePicker.CustomFormat** property to  
      "MMMM dd 'at' t:mm tt". If the "at" substring is not escaped, the result is a "June

06 aP 3:00PM" because the "t" character is read as the one-letter AM/PM format string (see the format string table below).

*ccc) DataBindings*

*ddd) DefaultImeMode*

*eee) DefaultSize*

*fff) ToString*

#### *Description*

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

*ggg) DesignMode*

*hhh) DisplayRectangle*

*iii) Disposing*

*jjj) Dock*

*kkk) DropDownAlign*

*lll) ToString*

#### *Description*

Gets or sets the alignment of the drop-down calendar on the date-time control.

The calendar drop-down can be aligned to the left or right of the control.

*mmm) Enabled*

*nnn) Events*

*ooo) Focused*

*ppp) Font*

*qqq) FontHeight*

*rrr) ForeColor*

*sss) ToString*

## *Description*

*ttt) Format*

*uuu) ToString*

[C#]      public      DateTimePickerFormat      Format      {get;      set;}

[C++] public: \_\_property DateTimePickerFormat get \_Format();public: \_\_property

void      set \_Format(DateTimePickerFormat);

[VB]      Public      Property      Format      As      DateTimePickerFormat

[JScript] public function get Format() : DateTimePickerFormat;public function set

Format(DateTimePickerFormat);

## *Description*

Gets or sets the format of the date and time displayed in the control.

This property determines the date-time format the date is displayed in. The date-time format is based on the user's regional settings in their operating system.



vvv) *Handle*

www) *HasChildren*

xxx) *Height*

yyy) *ImeMode*

zzz) *InvokeRequired*

aaaa) *IsAccessible*

bbbb) *IsDisposed*

cccc) *IsHandleCreated*

dddd) *Left*

eeee) *Location*

ffff) *MaxDate*

gggg) *ToString*

### *Description*

Gets or sets the maximum date and time that can be selected in the control.

hhhh) *MinDate*

iiii) *ToString*

[C#]            public            DateTime            MinDate            {get;            set;}

[C++]   public: \_\_property DateTime get\_MinDate();public: \_\_property void  
set\_MinDate(DateTime);

[VB]            Public            Property            MinDate            As            DateTime

[JScript]   public   function   get   MinDate()   :   DateTime;public   function   set

1 MinDate(DateTime);

3 *Description*

4 Gets or sets the minimum date and time that can be selected in the control.

5 *jjjj) Name*

6 *kkkk) Parent*

7 *llll) PreferredHeight*

8 *mmmm)ToString*

11 *Description*

12 Gets the preferred height of the date-time picker control.

13 The preferred height is the minimum height needed to accommodate the  
14 displayed text with the assigned **System.Drawing.Font** applied.

1        *nnnn) ProductName*

2        *oooo) ProductVersion*

3        *pppp) RecreatingHandle*

4        *qqqq) Region*

5        *rrrr) RenderRightToLeft*

6        *ssss) ResizeRedraw*

7        *tttt) Right*

8        *uuuu) RightToLeft*

9        *vvvv) ShowCheckBox*

10       *www)ToString*

11  
12  
13       *Description*

14       Gets or sets a value indicating whether a check box is displayed to the left of the  
15       selected date.

16       When **System.Windows.Forms.DateTimePicker.ShowCheckBox** is set to  
17       **true** a check box is displayed to the left of the date in the control. When the  
18       check box is checked, the date-time value can be updated . When the check box  
19       is empty, the date-time value is unable to be changed.

19       *xxxx) ShowFocusCues*

20       *yyyy) ShowKeyboardCues*

21       *zzzz) ShowUpDown*

22       *aaaaa) ToString*

23  
24  
25       *Description*

Gets or sets a value indicating whether an up-down control is used to adjust the date-time value.

When **System.Windows.Forms.DateTimePicker.ShowUpDown** is set to **true** , an up-down control is used to adjust time values instead of the drop-down calendar. The date-time can be adjusted by selecting each element individually and using the up and down buttons to change the value.

*bbbbbb) Site*

*cccccc) Size*

*dddddd) TabIndex*

*eeeeee) TabStop*

*ffffff) Tag*

*gggggg) Text*

*hhhhh)ToString*

#### *Description*

Gets or sets the text associated with this control.

The string returned by this property is equivalent to the **System.Windows.Forms.DateTimePicker.Value** property with the appropriate formatting or custom formatting applied. For example, if the **System.Windows.Forms.DateTimePicker.Value** property is set to 06/01/2001 12:00:00 AM while the **System.Windows.Forms.DateTimePicker.CustomFormat** property is set to "dddd, MMMM dd, yyyy", the **System.Windows.Forms.DateTimePicker.Text** property value is "Friday, June 01, 2001".

iiii) *Top*

jjjj) *TopLevelControl*

kkkk) *Value*

llll) *ToString*

*Description*

Gets or sets the date-time value assigned to the control.

If the **System.Windows.Forms.DateTimePicker.Value** property has not been changed in code or by the user, it is set to the current date and time ( **System.DateTime.Now** ).

mmmm) *Visible*

nnnn) *Width*

oooo) *WindowTarget*

pppp) *ToString*

*Description*

Occurs when the drop-down calendar is dismissed and disappears.

For more information about handling events, see .

qqqq) *ToString*

*Description*

Occurs when the drop-down calendar is shown.

For more information about handling events, see .

*rrrrr) ToString*

*Description*

Occurs when the **System.Windows.Forms.DateTimePicker.Format** property value has changed.

For more information about handling events, see .

*sssss) ToString*

*Description*

Occurs when the **System.Windows.Forms.DateTimePicker.Value** property changes.

For more information about handling events, see .

*ttttt) CreateAccessibilityInstance*

[C#]   protected   override   AccessibleObject   CreateAccessibilityInstance();

[C++]   protected:   AccessibleObject\*   CreateAccessibilityInstance();

[VB]   Overrides   Protected   Function   CreateAccessibilityInstance()   As  
AccessibleObject

[JScript]   protected   override   function   CreateAccessibilityInstance()   :  
AccessibleObject;

*Description*

*Description*

Constructs the new instance of the accessibility object for this control. Subclasses should not call base.CreateAccessibilityObject.

#### *uuuuu)CreateHandle*

|           |            |           |          |                 |
|-----------|------------|-----------|----------|-----------------|
| [C#]      | protected  | override  | void     | CreateHandle(); |
| [C++]     | protected: |           | void     | CreateHandle(); |
| [VB]      | Overrides  | Protected | Sub      | CreateHandle()  |
| [JScript] | protected  | override  | function | CreateHandle(); |

#### *Description*

Creates the physical window handle.

#### *vvvvv) DestroyHandle*

|           |            |           |          |                  |
|-----------|------------|-----------|----------|------------------|
| [C#]      | protected  | override  | void     | DestroyHandle(); |
| [C++]     | protected: |           | void     | DestroyHandle(); |
| [VB]      | Overrides  | Protected | Sub      | DestroyHandle()  |
| [JScript] | protected  | override  | function | DestroyHandle(); |

#### *Description*

Destroys the physical window handle.

#### *wwwww)IsInputKey*

|       |            |           |          |                  |                     |
|-------|------------|-----------|----------|------------------|---------------------|
| [C#]  | protected  | override  | bool     | IsInputKey(Keys  | keyData);           |
| [C++] | protected: |           | bool     | IsInputKey(Keys  | keyData);           |
| [VB]  | Overrides  | Protected | Function | IsInputKey(ByVal | keyData As Keys) As |

Boolean

[JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

#### *Description*

Determines whether the specified key is a regular input key or a special key that requires preprocessing.

*Return Value:* **true** if the specified key is a regular input key; otherwise, **false** .

Call this method during window-message preprocessing to determine whether the specified key is a regular input key that should be sent directly to the control or a special key (such as PAGE UP and PAGE DOWN) that should be preprocessed. In the latter case, send the key to the control only if it is not consumed by the preprocessing phase. One of the **System.Windows.Forms.Keys** values.

#### *xxxxx) OnCloseUp*

[C#] protected virtual void OnCloseUp(EventArgs eventargs);

[C++] protected: virtual void OnCloseUp(EventArgs\* eventargs);

[VB] Overridable Protected Sub OnCloseUp(ByVal eventargs As EventArgs)

[JScript] protected function OnCloseUp(eventargs : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DateTimePicker.CloseUp** event.

The **System.Windows.Forms.DateTimePicker.CloseUp** event occurs when the drop-down calendar is dismissed and disappears. An **System.EventArgs** that contains the event data.

#### *yyyyy) OnDropDown*

[C#] protected virtual void OnDropDown(EventArgs eventargs);

[C++] protected: virtual void OnDropDown(EventArgs\* eventargs);



1 [VB] Overridable Protected Sub OnDropDown(ByVal eventargs As EventArgs)

2 [JScript] protected function OnDropDown(eventargs : EventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.DateTimePicker.DropDown** event.

6 The **System.Windows.Forms.DateTimePicker.DropDown** event occurs  
7 when the drop-down calendar is shown. An **System.EventArgs** that contains  
the event data.

8 *zzzzz) OnFontChanged*

9  
10 [C#] protected override void OnFontChanged(EventArgs e);

11 [C++] protected: void OnFontChanged(EventArgs\* e);

12 [VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

13 [JScript] protected override function OnFontChanged(e : EventArgs);

14  
15 *Description*

16 Occurs when a property for the control changes. Event data

17 *aaaaaa)OnFormatChanged*

18  
19 [C#] protected virtual void OnFormatChanged(EventArgs e);

20 [C++] protected: virtual void OnFormatChanged(EventArgs\* e);

21 [VB] Overridable Protected Sub OnFormatChanged(ByVal e As EventArgs)

22 [JScript] protected function OnFormatChanged(e : EventArgs);

23  
24 *Description*

1 Raises the **System.Windows.Forms.DateTimePicker.FormatChanged** event.

2 The **System.Windows.Forms.DateTimePicker.FormatChanged** event  
3 occurs when the **System.Windows.Forms.DateTimePicker.Format** property  
4 value has changed. An **System.EventArgs** that contains the event data.

5 *bbbbbb)OnSystemColorsChanged*

6 [C#] protected override void OnSystemColorsChanged(EventArgs e);

7 [C++] protected: void OnSystemColorsChanged(EventArgs\* e);

8 [VB] Overrides Protected Sub OnSystemColorsChanged(ByVal e As EventArgs)

9 [JScript] protected override function OnSystemColorsChanged(e : EventArgs);

10  
11 *Description*

12 Handles system color changes Handles system color changes

13 *cccccc)OnValueChanged*

14  
15 [C#] protected virtual void OnValueChanged(EventArgs eventargs);

16 [C++] protected: virtual void OnValueChanged(EventArgs\* eventargs);

17 [VB] Overridable Protected Sub OnValueChanged(ByVal eventargs As  
18 EventArgs)

19 [JScript] protected function OnValueChanged(eventargs : EventArgs);

20  
21 *Description*

22 Raises the **System.Windows.Forms.DateTimePicker.ValueChanged** event.

23 The **System.Windows.Forms.DateTimePicker.ValueChanged** event occurs  
24 when the value for the control changes. An **System.EventArgs** that contains  
25 the event data.

### *dddddd)SetBoundsCore*

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);

[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)

[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

#### *Description*

Restricts the vertical size of the control Restricts the vertical size of the control

### *eeeeee)ToString*

[C#] public override string ToString();

[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

#### *Description*

Returns the control as a string

*Return Value:* String Returns the control as a string

*fffff) WndProc*

[C#]    protected    override    void    WndProc(ref    Message    m);  
[C++]        protected:        void        WndProc(Message\*        m);  
[VB]    Overrides    Protected    Sub    WndProc(ByRef    m    As    Message)  
[JScript]    protected    override    function    WndProc(m    :    Message);

*Description*

Overridden wndProc Overridden wndProc

DateTimePicker.DateTimePickerAccessibleObject                    class  
(System.Windows.Forms)

*a)    WndProc*

*Description*

*Description*

*b)    DateTimePicker.DateTimePickerAccessibleObject*

*Example Syntax:*

*c)    WndProc*

[C#]    public    DateTimePicker.DateTimePickerAccessibleObject(DateTimePicker  
owner);  
[C++]    public:    DateTimePickerAccessibleObject(DateTimePicker\*    owner);  
[VB]    Public    Sub    New(ByVal    owner    As    DateTimePicker)

1 [JScript] public function DateTimePicker.DateTimePickerAccessibleObject(owner  
2 : DateTimePicker);

3  
4 *Description*

- 5
- 6 *d) Bounds*
  - 7 *e) DefaultAction*
  - 8 *f) Description*
  - 9 *g) Handle*
  - 10 *h) Help*
  - 11 *i) KeyboardShortcut*
  - 12 *j) Name*
  - 13 *k) Owner*
  - 14 *l) Parent*
  - 15 *m) Role*
  - 16 *n) State*
  - 17 *o) WndProc*
- 18  
19  
20

21 *Description*  
22  
23  
24  
25

1        *p) Value*

2        *q) WndProc*

3  
4 [C#]        public        override        string        Value        {get;}

5 [C++]        public:        \_\_property        virtual        String\*        get\_Value();

6 [VB]        Overrides        Public        ReadOnly        Property        Value        As        String

7 [JScript]        public        function        get        Value()        :        String;

8  
9 *Description*

10  
11        DateTimePickerFormat enumeration (System.Windows.Forms)

12        *a) UseStdAccessibleObjects*

13  
14  
15 *Description*

16 Specifies the date and time format the  
**System.Windows.Forms.DateTimePicker** control displays.

17 This enumeration is used by members such as  
**System.Windows.Forms.DateTimePicker.Format** .

18  
19        *b) UseStdAccessibleObjects*

20  
21 [C#]        public        const        DateTimePickerFormat        Custom;

22 [C++]        public:        const        DateTimePickerFormat        Custom;

23 [VB]        Public        Const        Custom        As        DateTimePickerFormat

24 [JScript]        public        var        Custom        :        DateTimePickerFormat;

*Description*

The **System.Windows.Forms.DateTimePicker** control displays the date-time value in a custom format.

*c) UseStdAccessibleObjects*

|           |         |       |                      |                       |
|-----------|---------|-------|----------------------|-----------------------|
| [C#]      | public  | const | DateTimePickerFormat | Long;                 |
| [C++]     | public: | const | DateTimePickerFormat | Long;                 |
| [VB]      | Public  | Const | Long As              | DateTimePickerFormat  |
| [JScript] | public  | var   | Long :               | DateTimePickerFormat; |

*Description*

The **System.Windows.Forms.DateTimePicker** control displays the date-time value in the long date format set by the user's operating system.

*d) UseStdAccessibleObjects*

|           |         |       |                      |                       |
|-----------|---------|-------|----------------------|-----------------------|
| [C#]      | public  | const | DateTimePickerFormat | Short;                |
| [C++]     | public: | const | DateTimePickerFormat | Short;                |
| [VB]      | Public  | Const | Short As             | DateTimePickerFormat  |
| [JScript] | public  | var   | Short :              | DateTimePickerFormat; |

*Description*

The **System.Windows.Forms.DateTimePicker** control displays the date-time value in the short date format set by the user's operating system.

e) *UseStdAccessibleObjects*

|           |         |       |                      |                         |
|-----------|---------|-------|----------------------|-------------------------|
| [C#]      | public  | const | DateTimePickerFormat | Time;                   |
| [C++]     | public: | const | DateTimePickerFormat | Time;                   |
| [VB]      | Public  | Const | Time                 | As DateTimePickerFormat |
| [JScript] | public  | var   | Time                 | : DateTimePickerFormat; |

*Description*

The **System.Windows.Forms.DateTimePicker** control displays the date-time value in the time format set by the user's operating system.

Day enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies the day of the week.

This class is used by only **System.Windows.Forms.MonthCalendar** .

b) *ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Day     | Default; |
| [C++]     | public: | const | Day     | Default; |
| [VB]      | Public  | Const | Default | As Day   |
| [JScript] | public  | var   | Default | : Day;   |

*Description*

A default day of the week specified by the application.



c) *ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Day    | Friday; |
| [C++]     | public: | const | Day    | Friday; |
| [VB]      | Public  | Const | Friday | As Day  |
| [JScript] | public  | var   | Friday | : Day;  |

*Description*

The day Friday.

d) *ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Day    | Monday; |
| [C++]     | public: | const | Day    | Monday; |
| [VB]      | Public  | Const | Monday | As Day  |
| [JScript] | public  | var   | Monday | : Day;  |

*Description*

The day Monday.

e) *ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Day      | Saturday; |
| [C++]     | public: | const | Day      | Saturday; |
| [VB]      | Public  | Const | Saturday | As Day    |
| [JScript] | public  | var   | Saturday | : Day;    |

*Description*

The day Saturday.

*f) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Day    | Sunday; |
| [C++]     | public: | const | Day    | Sunday; |
| [VB]      | Public  | Const | Sunday | As Day  |
| [JScript] | public  | var   | Sunday | : Day;  |

*Description*

The day Sunday.

*g) ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Day      | Thursday; |
| [C++]     | public: | const | Day      | Thursday; |
| [VB]      | Public  | Const | Thursday | As Day    |
| [JScript] | public  | var   | Thursday | : Day;    |

*Description*

The day Thursday.

*h) ToString*

|       |         |       |     |          |
|-------|---------|-------|-----|----------|
| [C#]  | public  | const | Day | Tuesday; |
| [C++] | public: | const | Day | Tuesday; |

|           |        |       |         |    |      |
|-----------|--------|-------|---------|----|------|
| [VB]      | Public | Const | Tuesday | As | Day  |
| [JScript] | public | var   | Tuesday | :  | Day; |

*Description*

The day Tuesday.

*i) ToString*

|      |        |       |     |            |
|------|--------|-------|-----|------------|
| [C#] | public | const | Day | Wednesday; |
|------|--------|-------|-----|------------|

|       |         |       |     |            |
|-------|---------|-------|-----|------------|
| [C++] | public: | const | Day | Wednesday; |
|-------|---------|-------|-----|------------|

|      |        |       |           |    |     |
|------|--------|-------|-----------|----|-----|
| [VB] | Public | Const | Wednesday | As | Day |
|------|--------|-------|-----------|----|-----|

|           |        |     |           |   |      |
|-----------|--------|-----|-----------|---|------|
| [JScript] | public | var | Wednesday | : | Day; |
|-----------|--------|-----|-----------|---|------|

*Description*

The day Wednesday.

DialogResult enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies identifiers to indicate the return value of a dialog box.

The **System.Windows.Forms.Button.DialogResult** property and the **System.Windows.Forms.Form.ShowDialog** method use this enumeration.

*b) ToString*

|      |        |       |              |        |
|------|--------|-------|--------------|--------|
| [C#] | public | const | DialogResult | Abort; |
|------|--------|-------|--------------|--------|

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C++]     | public: | const | DialogResult | Abort;          |
| [VB]      | Public  | Const | Abort        | As DialogResult |
| [JScript] | public  | var   | Abort        | : DialogResult; |

*Description*

The dialog box return value is Abort (usually sent from a button labeled Abort).

*c) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | DialogResult | Cancel;         |
| [C++]     | public: | const | DialogResult | Cancel;         |
| [VB]      | Public  | Const | Cancel       | As DialogResult |
| [JScript] | public  | var   | Cancel       | : DialogResult; |

*Description*

The dialog box return value is Cancel (usually sent from a button labeled Cancel).

*d) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | DialogResult | Ignore;         |
| [C++]     | public: | const | DialogResult | Ignore;         |
| [VB]      | Public  | Const | Ignore       | As DialogResult |
| [JScript] | public  | var   | Ignore       | : DialogResult; |

*Description*

The dialog box return value is Ignore (usually sent from a button labeled Ignore).

**e) ToString**

|           |         |       |    |              |               |
|-----------|---------|-------|----|--------------|---------------|
| [C#]      | public  | const |    | DialogResult | No;           |
| [C++]     | public: | const |    | DialogResult | No;           |
| [VB]      | Public  | Const | No | As           | DialogResult  |
| [JScript] | public  | var   | No | :            | DialogResult; |

**Description**

The dialog box return value is No (usually sent from a button labeled No).

**f) ToString**

|           |         |       |      |              |               |
|-----------|---------|-------|------|--------------|---------------|
| [C#]      | public  | const |      | DialogResult | None;         |
| [C++]     | public: | const |      | DialogResult | None;         |
| [VB]      | Public  | Const | None | As           | DialogResult  |
| [JScript] | public  | var   | None | :            | DialogResult; |

**Description**

Nothing is returned from the dialog box. This means that the modal dialog continues running.

**g) ToString**

|           |         |       |    |              |               |
|-----------|---------|-------|----|--------------|---------------|
| [C#]      | public  | const |    | DialogResult | OK;           |
| [C++]     | public: | const |    | DialogResult | OK;           |
| [VB]      | Public  | Const | OK | As           | DialogResult  |
| [JScript] | public  | var   | OK | :            | DialogResult; |

## Description

The dialog box return value is OK (usually sent from a button labeled OK).

### *h) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | DialogResult | Retry;          |
| [C++]     | public: | const | DialogResult | Retry;          |
| [VB]      | Public  | Const | Retry        | As DialogResult |
| [JScript] | public  | var   | Retry        | : DialogResult; |

## Description

The dialog box return value is Retry (usually sent from a button labeled Retry).

### *i) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | DialogResult | Yes;            |
| [C++]     | public: | const | DialogResult | Yes;            |
| [VB]      | Public  | Const | Yes          | As DialogResult |
| [JScript] | public  | var   | Yes          | : DialogResult; |

## Description

The dialog box return value is Yes (usually sent from a button labeled Yes).

ScrollableControl.DockPaddingEdges class (System.Windows.Forms)

a) *ToString*

#### *Description*

Determines the border padding for docked controls.

The **System.Windows.Forms.ScrollableControl.DockPaddingEdges** class creates a margin on a given edge or all edges of a docked control. You may set the width of this margin for each individual edge by setting the following properties:

**System.Windows.Forms.ScrollableControl.DockPaddingEdges.Bottom** ,  
**System.Windows.Forms.ScrollableControl.DockPaddingEdges.Top** ,  
**System.Windows.Forms.ScrollableControl.DockPaddingEdges.Left** ,  
**System.Windows.Forms.ScrollableControl.DockPaddingEdges.Right** .

Alternatively, you may set all the edges to the same width simultaneously by setting the

**System.Windows.Forms.ScrollableControl.DockPaddingEdges.All** property. If the size of the control is too large for its container, the control will be resized to fit in the container, minus the specified margin width.

b) *All*

c) *ToString*

[C#]            public            int            All            {get;            set;}

[C++] public: \_\_property int get\_All();public: \_\_property void set\_All(int);

[VB]            Public            Property            All            As            Integer

[JScript] public function get All() : int;public function set All(int);

#### *Description*

Gets or sets the padding width for all edges of a docked control.

The padding width assigned to this property is applied to all edges of the docked control.

d) *Bottom*

e) *ToString*

[C#] public int Bottom {get; set;}

[C++] public: \_\_property int get\_Bottom();public: \_\_property void  
set\_Bottom(int);

[VB] Public Property Bottom As Integer

[JScript] public function get Bottom() : int;public function set Bottom(int);

#### *Description*

Gets or sets the padding width for the bottom edge of a docked control.

The padding width assigned to this property is applied only to the bottom edge of the docked control.

f) *Left*

g) *ToString*

[C#] public int Left {get; set;}

[C++] public: \_\_property int get\_Left();public: \_\_property void set\_Left(int);

[VB] Public Property Left As Integer

[JScript] public function get Left() : int;public function set Left(int);

#### *Description*

Gets or sets the padding width for the left edge of a docked control.

The padding width assigned to this property is applied only to the left edge of the docked control.



h) *Right*

i) *ToString*

[C#]            public            int            Right            {get;            set;}

[C++] public: \_\_property int get\_Right();public: \_\_property void set\_Right(int);

[VB]            Public            Property            Right            As            Integer

[JScript] public function get Right() : int;public function set Right(int);

#### *Description*

Gets or sets the padding width for the right edge of a docked control.

The padding width assigned to this property is applied only to the right edge of the docked control.

j) *Top*

k) *ToString*

[C#]            public            int            Top            {get;            set;}

[C++] public: \_\_property int get\_Top();public: \_\_property void set\_Top(int);

[VB]            Public            Property            Top            As            Integer

[JScript] public function get Top() : int;public function set Top(int);

#### *Description*

Gets or sets the padding width for the top edge of a docked control.

The padding width assigned to this property is applied only to the top edge of the docked control.

l) *Equals*

[C#] public override bool Equals(object other);  
[C++] public: bool Equals(Object\* other);  
[VB] Overrides Public Function Equals(ByVal other As Object) As Boolean  
[JScript] public override function Equals(other : Object) : Boolean;

*Description*

m) *GetHashCode*

[C#] public override int GetHashCode();  
[C++] public: int GetHashCode();  
[VB] Overrides Public Function GetHashCode() As Integer  
[JScript] public override function GetHashCode() : int;

*Description*

n) *ICloneable.Clone*

[C#] object ICloneable.Clone();  
[C++] Object\* ICloneable::Clone();  
[VB] Function Clone() As Object Implements ICloneable.Clone  
[JScript] function ICloneable.Clone() : Object;

***o) ToString***

```
1
2
3 [C#]          public          override          string          ToString();
4
5 [C++]          public:          String*          ToString();
6
7 [VB]  Overrides  Public  Function  ToString()  As  String
8
9 [JScript] public override function ToString() : String;
10
11     ScrollableControl.DockPaddingEdgesConverter          class
12
13 (System.Windows.Forms)
```

***a) ToString***

***b) ScrollableControl.DockPaddingEdgesConverter***

*Example Syntax:*

***c) ToString***

***d) GetProperties***

```
15 [C#]          public          override          PropertyDescriptorCollection
16 GetProperties(ITypeDescriptorContext  context,  object  value,  Attribute[]
17 attributes);
18
19 [C++]          public:          PropertyDescriptorCollection*
20 GetProperties(ITypeDescriptorContext*  context,  Object*  value,  Attribute*
21 attributes[]);
22
23 [VB]  Overrides  Public  Function  GetProperties(ByVal  context  As
24 ITypeDescriptorContext, ByVal value As Object, ByVal attributes() As Attribute)
25 As
26     PropertyDescriptorCollection
27
28 [JScript] public override function GetProperties(context : ITypeDescriptorContext,
29 value : Object, attributes : Attribute[]) : PropertyDescriptorCollection;
```

## Description

Retrieves the set of properties for this type. By default, a type has does not return any properties. An easy implementation of this method can just call `TypeDescriptor.GetProperties` for the correct data type.

**Return Value:** The set of properties that should be exposed for this data type. If no properties should be exposed, thsi may return null. The default implementation always returns null. A type descriptor through which additional context may be provided. The value of the object to get the properties for.

### e) *GetPropertiesSupported*

```
[C#] public override bool GetPropertiesSupported(ITypeDescriptorContext
context);
```

```
[C++] public: bool GetPropertiesSupported(ITypeDescriptorContext* context);
```

```
[VB] Overrides Public Function GetPropertiesSupported(ByVal context As
ITypeDescriptorContext) As Boolean
```

```
[JScript] public override function GetPropertiesSupported(context :
ITypeDescriptorContext) : Boolean;
```

## Description

Determines if this object supports properties. By default, this is false.

**Return Value:** Returns true if `GetProperties` should be called to find the properties of this object. A type descriptor through which additional context may be provided.

DockStyle enumeration (System.Windows.Forms)

### a) *ToString*

## Description

Specifies the position and manner in which a control is docked.

This enumeration is used by members such as  
**System.Windows.Forms.Control.Dock** .

*b) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | DockStyle | Bottom;      |
| [C++]     | public: | const | DockStyle | Bottom;      |
| [VB]      | Public  | Const | Bottom    | As DockStyle |
| [JScript] | public  | var   | Bottom    | : DockStyle; |

*Description*

The control's bottom edge is docked to the bottom of its containing control.

*c) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | DockStyle | Fill;        |
| [C++]     | public: | const | DockStyle | Fill;        |
| [VB]      | Public  | Const | Fill      | As DockStyle |
| [JScript] | public  | var   | Fill      | : DockStyle; |

*Description*

All the control's edges are docked to the all edges of its containing control and sized appropriately.

*d) ToString*

|       |         |       |           |       |
|-------|---------|-------|-----------|-------|
| [C#]  | public  | const | DockStyle | Left; |
| [C++] | public: | const | DockStyle | Left; |

|           |        |       |      |    |            |
|-----------|--------|-------|------|----|------------|
| [VB]      | Public | Const | Left | As | DockStyle  |
| [JScript] | public | var   | Left | :  | DockStyle; |

*Description*

The control's left edge is docked to the left edge of its containing control.

*e) ToString*

|           |         |       |      |           |            |
|-----------|---------|-------|------|-----------|------------|
| [C#]      | public  | const |      | DockStyle | None;      |
| [C++]     | public: | const |      | DockStyle | None;      |
| [VB]      | Public  | Const | None | As        | DockStyle  |
| [JScript] | public  | var   | None | :         | DockStyle; |

*Description*

The control is not docked.

*f) ToString*

|           |         |       |       |           |            |
|-----------|---------|-------|-------|-----------|------------|
| [C#]      | public  | const |       | DockStyle | Right;     |
| [C++]     | public: | const |       | DockStyle | Right;     |
| [VB]      | Public  | Const | Right | As        | DockStyle  |
| [JScript] | public  | var   | Right | :         | DockStyle; |

*Description*

The control's right edge is docked to the right edge of its containing control.

**g) ToString**

```
[C#]      public      const      DockStyle      Top;
[C++]     public:     const      DockStyle      Top;
[VB]      Public      Const      Top      As      DockStyle
[JScript] public      var      Top      :      DockStyle;
```

*Description*

The control's top edge is docked to the top of its containing control.

DomainUpDown.DomainItemAccessibleObject class  
(System.Windows.Forms)

**a) ToString**

*Description*

**b) DomainUpDown.DomainItemAccessibleObject**

*Example Syntax:*

**c) ToString**

```
[C#]      public      DomainUpDown.DomainItemAccessibleObject(string      name,
AccessibleObject      parent);
[C++]     public:     DomainItemAccessibleObject(String*      name, AccessibleObject*
parent);
[VB]      Public Sub New(ByVal name As String, ByVal parent As AccessibleObject)
[JScript] public function DomainUpDown.DomainItemAccessibleObject(name :
```

String, parent : AccessibleObject);

*Description*

*d) Bounds*

*e) DefaultAction*

*f) Description*

*g) Help*

*h) KeyboardShortcut*

*i) Name*

*j) ToString*

*Description*

*k) Parent*

*l) ToString*

[C#] public override AccessibleObject Parent {get;}

[C++] public: \_\_property virtual AccessibleObject\* get\_Parent();

[VB] Overrides Public ReadOnly Property Parent As AccessibleObject

[JScript] public function get Parent() : AccessibleObject;

*Description*



1  
2        *m)    Role*

3        *n)    ToString*

4  
5 [C#]        public        override        AccessibleRole        Role        {get;}

6 [C++]        public:        \_\_property        virtual        AccessibleRole        get\_Role();

7 [VB]        Overrides        Public        ReadOnly        Property        Role        As        AccessibleRole

8 [JScript]        public        function        get        Role()        :        AccessibleRole;

9  
10 *Description*

11  
12        *o)    State*

13        *p)    ToString*

14  
15 [C#]        public        override        AccessibleStates        State        {get;}

16 [C++]        public:        \_\_property        virtual        AccessibleStates        get\_State();

17 [VB]        Overrides        Public        ReadOnly        Property        State        As        AccessibleStates

18 [JScript]        public        function        get        State()        :        AccessibleStates;

19  
20 *Description*

q) *Value*

r) *ToString*

|           |           |            |          |          |                 |
|-----------|-----------|------------|----------|----------|-----------------|
| [C#]      | public    | override   | string   | Value    | {get;}          |
| [C++]     | public:   | __property | virtual  | String*  | get_Value();    |
| [VB]      | Overrides | Public     | ReadOnly | Property | Value As String |
| [JScript] | public    | function   | get      | Value()  | : String;       |

#### *Description*

DomainUpDown class (System.Windows.Forms)

a) *UseStdAccessibleObjects*

#### *Description*

Represents a Windows up-down control that displays string values.

A **System.Windows.Forms.DomainUpDown** control displays a single string value that is selected from an **System.Object** collection by clicking the up or down buttons of the control. The user can also enter text in the control, unless the **System.Windows.Forms.UpDownBase.ReadOnly** property is set to **true** (the string typed in must match an item in the collection to be accepted). When an item is selected, the object is converted to a string value so it may be displayed in the up-down control.

b) *DomainUpDown*

*Example Syntax:*

c) *UseStdAccessibleObjects*

|           |         |                          |
|-----------|---------|--------------------------|
| [C#]      | public  | DomainUpDown();          |
| [C++]     | public: | DomainUpDown();          |
| [VB]      | Public  | Sub New()                |
| [JScript] | public  | function DomainUpDown(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.DomainUpDown** class.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *ActiveControl*
- j) *AllowDrop*
- k) *Anchor*
- l) *AutoScroll*
- m) *AutoScrollMargin*
- n) *AutoScrollMinSize*
- o) *AutoScrollPosition*
- p) *BackColor*
- q) *BackgroundImage*
- r) *BindingContext*
- s) *BorderStyle*
- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ChangingText*

|    |            |                                |
|----|------------|--------------------------------|
| 1  | <b>aa)</b> | <b><i>ClientRectangle</i></b>  |
| 2  | <b>bb)</b> | <b><i>ClientSize</i></b>       |
| 3  | <b>cc)</b> | <b><i>CompanyName</i></b>      |
| 4  | <b>dd)</b> | <b><i>Container</i></b>        |
| 5  | <b>ee)</b> | <b><i>ContainsFocus</i></b>    |
| 6  | <b>ff)</b> | <b><i>ContextMenu</i></b>      |
| 7  | <b>gg)</b> | <b><i>Controls</i></b>         |
| 8  | <b>hh)</b> | <b><i>Created</i></b>          |
| 9  | <b>ii)</b> | <b><i>CreateParams</i></b>     |
| 10 | <b>jj)</b> | <b><i>Cursor</i></b>           |
| 11 | <b>kk)</b> | <b><i>DataBindings</i></b>     |
| 12 | <b>ll)</b> | <b><i>DefaultImeMode</i></b>   |
| 13 | <b>mm)</b> | <b><i>DefaultSize</i></b>      |
| 14 | <b>nn)</b> | <b><i>DesignMode</i></b>       |
| 15 | <b>oo)</b> | <b><i>DisplayRectangle</i></b> |
| 16 | <b>pp)</b> | <b><i>Disposing</i></b>        |
| 17 | <b>qq)</b> | <b><i>Dock</i></b>             |
| 18 | <b>rr)</b> | <b><i>DockPadding</i></b>      |
| 19 | <b>ss)</b> | <b><i>Enabled</i></b>          |
| 20 | <b>tt)</b> | <b><i>Events</i></b>           |
| 21 | <b>uu)</b> | <b><i>Focused</i></b>          |
| 22 | <b>vv)</b> | <b><i>Font</i></b>             |
| 23 | <b>ww)</b> | <b><i>FontHeight</i></b>       |

- xx) *ForeColor*
- yy) *Handle*
- zz) *HasChildren*
- aaa) *Height*
- bbb) *HScroll*
- ccc) *ImeMode*
- ddd) *InterceptArrowKeys*
- eee) *InvokeRequired*
- fff) *IsAccessible*
- ggg) *IsDisposed*
- hhh) *IsHandleCreated*
- iii) *Items*
- jjj) *UseStdAccessibleObjects*

### *Description*

A collection of objects assigned to the up-down control.

The **System.Object** collection may be built and made available to the **System.Windows.Forms.DomainUpDown** control in two ways. You can add items to the collection by using the **System.Collections.ArrayList.Add(System.Object)** or **System.Collections.ArrayList.Insert(System.Int32, System.Object)** methods.

*kkk) Left*  
*lll) Location*  
*mmm) Name*  
*nnn) Parent*  
*ooo) ParentForm*  
*ppp) PreferredHeight*  
*qqq) ProductName*  
*rrr) ProductVersion*  
*sss) ReadOnly*  
*ttt) RecreatingHandle*  
*uuu) Region*  
*vvv) RenderRightToLeft*  
*www) ResizeRedraw*  
*xxx) Right*  
*yyy) RightToLeft*  
*zzz) SelectedIndex*  
*aaaa) UseStdAccessibleObjects*

## *Description*

Gets or sets the index value of the selected item.

The **System.Windows.Forms.DomainUpDown.SelectedIndex** property holds the index value of the item in the collection that is currently selected in the up-down control. Collection items may be re-assigned new index values if the **System.Windows.Forms.DomainUpDown.Sorted** property has been

changed from **false** to **true** . As the collection is re-sorted alphabetically, the items will be assigned a new index value.

*bbbb) SelectedItem*

*cccc) UseStdAccessibleObjects*

[C#]            public            object            SelectedItem            {get;            set;}

[C++] public: \_\_property Object\* get\_SelectedItem();public: \_\_property void  
set\_SelectedItem(Object\*);

[VB]            Public            Property            SelectedItem            As            Object

[JScript] public function get SelectedItem() : Object;public function set  
SelectedItem(Object);

### *Description*

Gets or sets the selected item based on the index value of the selected item in the collection.

When this property is set, the value is validated to be one of the items in the collection, and the

**System.Windows.Forms.DomainUpDown.SelectedIndex** property is set to the appropriate index value.



1        *dddd) ShowFocusCues*

2        *eeee) ShowKeyboardCues*

3        *ffff) Site*

4        *gggg) Size*

5        *hhhh) Sorted*

6        *iiii) UseStdAccessibleObjects*

7  
8  
9        *Description*

10       Gets or sets a value indicating whether the item collection is sorted.

11       When **System.Windows.Forms.DomainUpDown.Sorted** is set to **true** , the  
12       collection is sorted in alphabetical order.

13

14

15

16

17

18

19

20

21

22

23

24

25

1        *jjjj) TabIndex*

2        *kkkk) TabStop*

3        *lll) Tag*

4        *mmmm)Text*

5        *nnnn) TextAlign*

6        *oooo) Top*

7        *pppp) TopLevelControl*

8        *qqqq) UpDownAlign*

9        *rrrr) UserEdit*

10       *ssss) Visible*

11       *ttt) VScroll*

12       *uuuu) Width*

13       *vvvv) WindowTarget*

14       *www)Wrap*

15       *xxxx) UseStdAccessibleObjects*

16  
17  
18  
19       *Description*

20       Gets or sets a value indicating whether the collection of items continues to the  
21       first or last item if the user continues past the end of the list.

22       When **System.Windows.Forms.DomainUpDown.Wrap** is set to **true** , if you  
23       reach the last item in the collection and continue, the list will start over with the  
24       first item and appear to be continuous. This behavior works in reverse as well.  
25

yyyy) *UseStdAccessibleObjects*

*Description*

Occurs when the **System.Windows.Forms.DomainUpDown.SelectedItem** property has been changed.

For the **System.Windows.Forms.DomainUpDown.SelectedItemChanged** event to occur, the **System.Windows.Forms.DomainUpDown.SelectedItem** property can be changed in code, by the user typing in a new value or clicking the control's up or down buttons.

zzzz) *CreateAccessibilityInstance*

```
[C#]    protected    override    AccessibleObject    CreateAccessibilityInstance();
[C++]    protected:    AccessibleObject*    CreateAccessibilityInstance();
[VB]    Overrides    Protected    Function    CreateAccessibilityInstance() As
AccessibleObject
[JScript]    protected    override    function    CreateAccessibilityInstance() :
AccessibleObject;
```

*Description*

Constructs the new instance of the accessibility object for this control. Subclasses should not call base.CreateAccessibilityObject.

aaaaa) *DownButton*

```
[C#]    public    override    void    DownButton();
[C++]    public:    void    DownButton();
[VB]    Overrides    Public    Sub    DownButton()
```

| Variable   | Unit     | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 | 2096 | 2097 | 2098 | 2099 | 2100 |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------------|----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| Population | millions | 10.0 | 10.1 | 10.2 | 10.3 | 10.4 | 10.5 | 10.6 | 10.7 | 10.8 | 10.9 | 11.0 | 11.1 | 11.2 | 11.3 | 11.4 | 11.5 | 11.6 | 11.7 | 11.8 | 11.9 | 12.0 | 12.1 | 12.2 | 12.3 | 12.4 | 12.5 | 12.6 | 12.7 | 12.8 | 12.9 | 13.0 | 13.1 | 13.2 | 13.3 | 13.4 | 13.5 | 13.6 | 13.7 | 13.8 | 13.9 | 14.0 | 14.1 | 14.2 | 14.3 | 14.4 | 14.5 | 14.6 | 14.7 | 14.8 | 14.9 | 15.0 | 15.1 | 15.2 | 15.3 | 15.4 | 15.5 | 15.6 | 15.7 | 15.8 | 15.9 | 16.0 | 16.1 | 16.2 | 16.3 | 16.4 | 16.5 | 16.6 | 16.7 | 16.8 | 16.9 | 17.0 | 17.1 | 17.2 | 17.3 | 17.4 | 17.5 | 17.6 | 17.7 | 17.8 | 17.9 | 18.0 | 18.1 | 18.2 | 18.3 | 18.4 | 18.5 | 18.6 | 18.7 | 18.8 | 18.9 | 19.0 | 19.1 | 19.2 | 19.3 | 19.4 | 19.5 | 19.6 | 19.7 | 19.8 | 19.9 | 20.0 | 20.1 | 20.2 | 20.3 | 20.4 | 20.5 | 20.6 | 20.7 | 20.8 | 20.9 | 21.0 | 21.1 | 21.2 | 21.3 | 21.4 | 21.5 | 21.6 | 21.7 | 21.8 | 21.9 | 22.0 | 22.1 | 22.2 | 22.3 | 22.4 | 22.5 | 22.6 | 22.7 | 22.8 | 22.9 | 23.0 | 23.1 | 23.2 | 23.3 | 23.4 | 23.5 | 23.6 | 23.7 | 23.8 | 23.9 | 24.0 | 24.1 | 24.2 | 24.3 | 24.4 | 24.5 | 24.6 | 24.7 | 24.8 | 24.9 | 25.0 | 25.1 | 25.2 | 25.3 | 25.4 | 25.5 | 25.6 | 25.7 | 25.8 | 25.9 | 26.0 | 26.1 | 26.2 | 26.3 | 26.4 | 26.5 | 26.6 | 26.7 | 26.8 | 26.9 | 27.0 | 27.1 | 27.2 | 27.3 | 27.4 | 27.5 | 27.6 | 27.7 | 27.8 | 27.9 | 28.0 | 28.1 | 28.2 | 28.3 | 28.4 | 28.5 | 28.6 | 28.7 | 28.8 | 28.9 | 29.0 | 29.1 | 29.2 | 29.3 | 29.4 | 29.5 | 29.6 | 29.7 | 29.8 | 29.9 | 30.0 | 30.1 | 30.2 | 30.3 | 30.4 | 30.5 | 30.6 | 30.7 | 30.8 | 30.9 | 31.0 | 31.1 | 31.2 | 31.3 | 31.4 | 31.5 | 31.6 | 31.7 | 31.8 | 31.9 | 32.0 | 32.1 | 32.2 | 32.3 | 32.4 | 32.5 | 32.6 | 32.7 | 32.8 | 32.9 | 33.0 | 33.1 | 33.2 | 33.3 | 33.4 | 33.5 | 33.6 | 33.7 | 33.8 | 33.9 | 34.0 | 34.1 | 34.2 | 34.3 | 34.4 | 34.5 | 34.6 | 34.7 | 34.8 | 34.9 | 35.0 | 35.1 | 35.2 | 35.3 | 35.4 | 35.5 | 35.6 | 35.7 | 35.8 | 35.9 | 36.0 | 36.1 | 36.2 | 36.3 | 36.4 | 36.5 | 36.6 | 36.7 | 36.8 | 36.9 | 37.0 | 37.1 | 37.2 | 37.3 | 37.4 | 37.5 | 37.6 | 37.7 | 37.8 | 37.9 | 38.0 | 38.1 | 38.2 | 38.3 | 38.4 | 38.5 | 38.6 | 38.7 | 38.8 | 38.9 | 39.0 | 39.1 | 39.2 |

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

[JScript] protected function OnSelectedItemChanged(source : Object, e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.DomainUpDown.SelectedItemChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . The source of the event. An **System.EventArgs** that contains the event data.

#### *dddd) OnTextBoxKeyDown*

[C#] protected override void OnTextBoxKeyDown(object source, KeyEventArgs e);

[C++] protected: void OnTextBoxKeyDown(Object\* source, KeyEventArgs\* e);

[VB] Overrides Protected Sub OnTextBoxKeyDown(ByVal source As Object, ByVal e As KeyEventArgs)

[JScript] protected override function OnTextBoxKeyDown(source : Object, e : KeyEventArgs);

#### *Description*

DEV: Overridden OnTextBoxKeyDown - when the user types a key in a read-only domain updown, the first/next item in the domain list is selected.

#### *eeee) ToString*

[C#] public override string ToString();

[C++] public: String\* ToString();

|           |           |          |          |            |    |         |
|-----------|-----------|----------|----------|------------|----|---------|
| [VB]      | Overrides | Public   | Function | ToString() | As | String  |
| [JScript] | public    | override | function | ToString() | :  | String; |

#### Description

Provides some interesting info about this control in String form.

*Return Value:* String Provides some interesting info about this control in String form.

#### fffff) UpButton

|           |           |          |          |             |
|-----------|-----------|----------|----------|-------------|
| [C#]      | public    | override | void     | UpButton(); |
| [C++]     | public:   |          | void     | UpButton(); |
| [VB]      | Overrides | Public   | Sub      | UpButton()  |
| [JScript] | public    | override | function | UpButton(); |

#### Description

Displays the previous item in the collection.

As you move through the collection of items in the

**System.Windows.Forms.DomainUpDown** control using the up button, you will eventually reach the first item in the collection. If you continue, and

**System.Windows.Forms.DomainUpDown.Wrap** is set to **true**, the list will start over with the last item in the collection and appear to be continuous. This behavior is also true when moving through the collection using the down button.

#### ggggg) UpdateEditText

|           |            |           |          |                   |
|-----------|------------|-----------|----------|-------------------|
| [C#]      | protected  | override  | void     | UpdateEditText(); |
| [C++]     | protected: |           | void     | UpdateEditText(); |
| [VB]      | Overrides  | Protected | Sub      | UpdateEditText()  |
| [JScript] | protected  | override  | function | UpdateEditText(); |

1  
2 *Description*

3 Updates the text in the up-down control to display the selected item.

4 DomainUpDown.DomainUpDownAccessibleObject class  
5 (System.Windows.Forms)

6 a) *WndProc*

7  
8  
9 *Description*

10 b) *DomainUpDown.DomainUpDownAccessibleObject*

11 *Example Syntax:*

12 c) *WndProc*

13  
14 [C#] public DomainUpDown.DomainUpDownAccessibleObject(Control owner);  
15 [C++] public: DomainUpDownAccessibleObject(Control\* owner);  
16 [VB] Public Sub New(ByVal owner As Control)  
17 [JScript] public function  
18 DomainUpDown.DomainUpDownAccessibleObject(owner : Control);  
19

20 *Description*  
21  
22  
23  
24  
25

- d) *Bounds*
- e) *DefaultAction*
- f) *Description*
- g) *Handle*
- h) *Help*
- i) *KeyboardShortcut*
- j) *Name*
- k) *Owner*
- l) *Parent*
- m) *Role*
- n) *WndProc*

#### *Description*

- o) *State*
- p) *Value*
- q) *GetChild*

[C#]     public     override     AccessibleObject     GetChild(int     index);

[C++]     public:     AccessibleObject\*     GetChild(int     index);

[VB]     Overrides     Public     Function     GetChild(ByVal     index     As     Integer)     As  
AccessibleObject

[JScript]     public     override     function     GetChild(index : int) : AccessibleObject;



*Description*

*r) GetChildCount*

```
[C#]      public      override      int      GetChildCount();
[C++]      public:      int      GetChildCount();
[VB]      Overrides      Public      Function      GetChildCount()      As      Integer
[JScript]      public      override      function      GetChildCount()      :      int;
```

*Description*

DomainUpDown.DomainUpDownItemCollection class  
(System.Windows.Forms)

*a) UseStdAccessibleObjects*

*Description*

Encapsulates a collection of objects for use by the **System.Windows.Forms.DomainUpDown** class.

To create a collection of objects to display in the **System.Windows.Forms.DomainUpDown** control, you can add or remove the items individually by using the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.Add(System.Object)** and **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.Remove(System.Object)** methods. The collection is accessed from the parent control, **System.Windows.Forms.DomainUpDown**, by the **System.Windows.Forms.DomainUpDown.Items** property.

- b) *Capacity*
- c) *Count*
- d) *IsFixedSize*
- e) *IsReadOnly*
- f) *IsSynchronized*
- g) *Item*
- h) *UseStdAccessibleObjects*

#### *Description*

Gets or sets the item at the specified indexed location in the collection.

To assign items to a specific location, or to retrieve them from the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection**, you can reference the collection object with a specific index value. The index value of the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection** is a zero-based index. The indexed location of the item in the collection.

i) *SyncRoot*

j) *Add*

[C#]            public            override            int            Add(object            item);

[C++]            public:            int            Add(Object\*            item);

[VB] Overrides Public Function Add(ByVal item As Object) As Integer

[JScript]    public    override    function    Add(item    :    Object)    :    int;

#### *Description*

Adds a the specified object to the end of the collection.

*Return Value:* The zero-based index value of the **System.Object** added to the collection.

You can also add a new **System.Object** to the collection by using the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.Insert(System.Int32,System.Object)** method. The **System.Object** to be added to the end of the collection.

#### *k) Insert*

[C#] public override void Insert(int index, object item);

[C++] public: void Insert(int index, Object\* item);

[VB] Overrides Public Sub Insert(ByVal index As Integer, ByVal item As Object)

[JScript] public override function Insert(index : int, item : Object);

#### *Description*

Inserts an existing toolbar button in the toolbar button collection at the specified location.

You can also add a new **System.Object** to the collection by using the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.Add(System.Object)** method. The indexed location within the collection to insert the **System.Object** . The **System.Object** to insert.

#### *l) Remove*

[C#] public override void Remove(object item);

[C++] public: void Remove(Object\* item);

[VB] Overrides Public Sub Remove(ByVal item As Object)

[JScript] public override function Remove(item : Object);

#### *Description*

Removes the specified item from the collection.

You can also remove an **System.Object** that you have previously added by using the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.RemoveAt(System.Int32)** method. The **System.Object** to remove from the collection.

*m) RemoveAt*

[C#]        public        override        void        RemoveAt(int        item);

[C++]        public:        void        RemoveAt(int        item);

[VB]    Overrides    Public    Sub    RemoveAt(ByVal    item    As    Integer)

[JScript]    public    override    function    RemoveAt(item    :    int);

*Description*

Removes the item from the specified location in the collection.

You can also remove an **System.Object** that you have previously added by using the **System.Windows.Forms.DomainUpDown.DomainUpDownItemCollection.Remove(System.Object)** method. The indexed location of the **System.Object** in the collection.

DragAction enumeration (System.Windows.Forms)

*a) TrimToSize*

*Description*

Specifies how and if a drag-and-drop operation should continue.

This enumeration is used by **System.Windows.Forms.QueryContinueDragEventArgs**.

**b) TrimToSize**

|           |         |       |            |               |
|-----------|---------|-------|------------|---------------|
| [C#]      | public  | const | DragAction | Cancel;       |
| [C++]     | public: | const | DragAction | Cancel;       |
| [VB]      | Public  | Const | Cancel     | As DragAction |
| [JScript] | public  | var   | Cancel     | : DragAction; |

*Description*

The operation is canceled with no drop message.

**c) TrimToSize**

|           |         |       |            |               |
|-----------|---------|-------|------------|---------------|
| [C#]      | public  | const | DragAction | Continue;     |
| [C++]     | public: | const | DragAction | Continue;     |
| [VB]      | Public  | Const | Continue   | As DragAction |
| [JScript] | public  | var   | Continue   | : DragAction; |

*Description*

The operation will continue.

**d) TrimToSize**

|           |         |       |            |               |
|-----------|---------|-------|------------|---------------|
| [C#]      | public  | const | DragAction | Drop;         |
| [C++]     | public: | const | DragAction | Drop;         |
| [VB]      | Public  | Const | Drop       | As DragAction |
| [JScript] | public  | var   | Drop       | : DragAction; |

1  
2 *Description*

3 The operation will stop with a drop.

4 DragDropEffects enumeration (System.Windows.Forms)

5 *a) ToString*

6  
7  
8 *Description*

9 Specifies the effects of a drag-and-drop operation.

10 This enumeration is used by the following classes:  
11 **System.Windows.Forms.DragEventArgs** ,  
12 **System.Windows.Forms.GiveFeedbackEventArgs** , and  
13 **System.Windows.Forms.Control** .

14  
15  
16 *b) ToString*

|              |         |       |                 |                  |
|--------------|---------|-------|-----------------|------------------|
| 17 [C#]      | public  | const | DragDropEffects | All;             |
| 18 [C++]     | public: | const | DragDropEffects | All;             |
| 19 [VB]      | Public  | Const | All As          | DragDropEffects  |
| 20 [JScript] | public  | var   | All :           | DragDropEffects; |

21  
22 *Description*

23 The data is copied, removed from the drag source, and scrolled in the drop target.

24  
25 *c) ToString*

|      |        |       |                 |       |
|------|--------|-------|-----------------|-------|
| [C#] | public | const | DragDropEffects | Copy; |
|------|--------|-------|-----------------|-------|

|           |         |       |                 |                  |
|-----------|---------|-------|-----------------|------------------|
| [C++]     | public: | const | DragDropEffects | Copy;            |
| [VB]      | Public  | Const | Copy As         | DragDropEffects  |
| [JScript] | public  | var   | Copy :          | DragDropEffects; |

*Description*

The data is copied to the drop target.

*d) ToString*

|           |         |       |                 |                  |
|-----------|---------|-------|-----------------|------------------|
| [C#]      | public  | const | DragDropEffects | Link;            |
| [C++]     | public: | const | DragDropEffects | Link;            |
| [VB]      | Public  | Const | Link As         | DragDropEffects  |
| [JScript] | public  | var   | Link :          | DragDropEffects; |

*Description*

The data from the drag source is linked to the drop target.

*e) ToString*

|           |         |       |                 |                  |
|-----------|---------|-------|-----------------|------------------|
| [C#]      | public  | const | DragDropEffects | Move;            |
| [C++]     | public: | const | DragDropEffects | Move;            |
| [VB]      | Public  | Const | Move As         | DragDropEffects  |
| [JScript] | public  | var   | Move :          | DragDropEffects; |

*Description*

The data from the drag source is moved to the drop target.

*f) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | DragDropEffects | None;              |
| [C++]     | public: | const | DragDropEffects | None;              |
| [VB]      | Public  | Const | None            | As DragDropEffects |
| [JScript] | public  | var   | None            | : DragDropEffects; |

*Description*

The drop target does not accept the data.

*g) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | DragDropEffects | Scroll;            |
| [C++]     | public: | const | DragDropEffects | Scroll;            |
| [VB]      | Public  | Const | Scroll          | As DragDropEffects |
| [JScript] | public  | var   | Scroll          | : DragDropEffects; |

*Description*

Scrolling is about to start or is currently occurring in the drop target.

DragEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **System.Windows.Forms.Control.DragDrop** , **System.Windows.Forms.Control.DragEnter** , or **System.Windows.Forms.Control.DragOver** event.



The **System.Windows.Forms.Control.DragDrop** event occurs when the user completes a drag-and-drop operation by dragging an object over the control and then dropping it onto the control by releasing the mouse button. The **System.Windows.Forms.Control.DragEnter** event occurs when the user moves the mouse pointer onto the control while dragging an object with the mouse. The **System.Windows.Forms.Control.DragOver** event occurs when the user moves the mouse pointer over the control while dragging an object with the mouse.

*b) DragEventArgs*

*Example Syntax:*

*c) ToString*

```
[C#] public DragEventArgs(IDataObject data, int keyState, int x, int y,
DragDropEffects allowedEffect, DragDropEffects effect);
[C++] public: DragEventArgs(IDataObject* data, int keyState, int x, int y,
DragDropEffects allowedEffect, DragDropEffects effect);
[VB] Public Sub New(ByVal data As IDataObject, ByVal keyState As Integer,
ByVal x As Integer, ByVal y As Integer, ByVal allowedEffect As
DragDropEffects, ByVal effect As DragDropEffects)
[JavaScript] public function DragEventArgs(data : IDataObject, keyState : int, x : int,
y : int, allowedEffect : DragDropEffects, effect : DragDropEffects);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.DragEventArgs** class. The data associated with this event. The current state of the SHIFT, CTRL, and ALT keys. The x-coordinate of the mouse cursor in pixels. The y-coordinate of the mouse cursor in pixels. One of the **System.Windows.Forms.DragDropEffects** values. One of the **System.Windows.Forms.DragDropEffects** values.

d) *AllowedEffect*

e) *ToString*

```
[C#]      public      DragDropEffects      AllowedEffect      {get;}
[C++]     public:     __property      DragDropEffects      get_AllowedEffect();
[VB]      Public      ReadOnly      Property      AllowedEffect      As      DragDropEffects
[JScript] public      function      get      AllowedEffect()      :      DragDropEffects;
```

*Description*

Gets which drag-and-drop operations are allowed by the originator (or source) of the drag event.

For example, when you start dragging a file from a source, if the file is read-only (or from a read-only storage medium such as a CD), the source will indicate that the file can be copied, but not transferred, to the target.

f) *Data*

g) *ToString*

```
[C#]      public      IDataObject      Data      {get;}
[C++]     public:     __property      IDataObject*      get_Data();
[VB]      Public      ReadOnly      Property      Data      As      IDataObject
[JScript] public      function      get      Data()      :      IDataObject;
```

*Description*

Gets the **System.Windows.Forms.IDataObject** that contains the data associated with this event.

h) *Effect*

i) *ToString*

```
[C#]      public      DragDropEffects      Effect      {get;      set;}
[C++] public: __property DragDropEffects get_Effect();public: __property void
set_Effect(DragDropEffects);
[VB]      Public      Property      Effect      As      DragDropEffects
[JScript] public function get Effect() : DragDropEffects;public function set
Effect(DragDropEffects);
```

#### *Description*

Gets or sets which drag-and-drop operations are allowed by the target of the drag event.

For example, when you drag a file from a source, if you press the CTRL key, the target will try to make it a copy operation.

j) *KeyState*

k) *ToString*

```
[C#]      public      int      KeyState      {get;}
[C++] public:      __property      int      get_KeyState();
[VB]      Public      ReadOnly      Property      KeyState      As      Integer
[JScript] public      function      get      KeyState()      :      int;
```

#### *Description*

Gets the current state of the SHIFT, CTRL, and ALT keys.

l) *X*

m) *ToString*

[C#] public int X {get;}

[C++] public: \_\_property int get\_X();

[VB] Public ReadOnly Property X As Integer

[JScript] public function get X() : int;

*Description*

Gets the x-coordinate of the mouse pointer.

n) *Y*

o) *ToString*

[C#] public int Y {get;}

[C++] public: \_\_property int get\_Y();

[VB] Public ReadOnly Property Y As Integer

[JScript] public function get Y() : int;

*Description*

Gets the y-coordinate of the mouse pointer.

DragEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **System.Windows.Forms.Control.DragDrop** , **System.Windows.Forms.Control.DragEnter** , or **System.Windows.Forms.Control.DragOver** event of a **System.Windows.Forms.Control** . The source of the event. A **System.Windows.Forms.DragEventArgs** that contains the event data.

When you create a **System.Windows.Forms.DragEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

DrawItemEventArgs class (System.Windows.Forms)

a) *ToString*

### *Description*

Provides data for the **DrawItem** event.

The **DrawItem** event is raised by owner draw controls, such as **System.Windows.Forms.ListBox** and **System.Windows.Forms.ComboBox** controls. It contains all the information needed for the user to paint the specified item, including the item index, the **System.Drawing.Rectangle** , and the **System.Drawing.Graphics** object on which the drawing should be done.

b) *DrawItemEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#] public DrawItemEventArgs(Graphics graphics, Font font, Rectangle rect, int
index, DrawItemState state);
[C++] public: DrawItemEventArgs(Graphics* graphics, Font* font, Rectangle
rect, int index, DrawItemState state);
```

```

1 [VB] Public Sub New(ByVal graphics As Graphics, ByVal font As Font, ByVal
2 rect As Rectangle, ByVal index As Integer, ByVal state As DrawItemState)
3 [JScript] public function DrawItemEventArgs(graphics : Graphics, font : Font, rect
4 : Rectangle, index : int, state : DrawItemState); Initializes a new instance of the
5 System.Windows.Forms.DrawItemEventArgs class.

```

### *Description*

Initializes a new instance of the **System.Windows.Forms.DrawItemEventArgs** class for the specified control with the specified font, state, surface to draw on, and the bounds to draw within. The **System.Drawing.Graphics** surface on which to draw. The **System.Drawing.Font** to use, usually the parent control's **System.Drawing.Font** property. The **System.Drawing.Rectangle** bounds to draw within. The **System.Windows.Forms.Control.ControlCollection** index value of the item that is being drawn. The control's **System.Windows.Forms.DrawItemState** information.

#### *d) DrawItemEventArgs*

#### *Example Syntax:*

#### *e) ToString*

```

17 [C#] public DrawItemEventArgs(Graphics graphics, Font font, Rectangle rect, int
18 index, DrawItemState state, Color foreColor, Color backColor);
19 [C++] public: DrawItemEventArgs(Graphics* graphics, Font* font, Rectangle
20 rect, int index, DrawItemState state, Color foreColor, Color backColor);
21 [VB] Public Sub New(ByVal graphics As Graphics, ByVal font As Font, ByVal
22 rect As Rectangle, ByVal index As Integer, ByVal state As DrawItemState, ByVal
23 foreColor As Color, ByVal backColor As Color)
24 [JScript] public function DrawItemEventArgs(graphics : Graphics, font : Font, rect
25

```

: Rectangle, index : int, state : DrawItemState, foreColor : Color, backColor : Color);

#### Description

Initializes a new instance of the **System.Windows.Forms.DrawItemEventArgs** class for the specified control with the specified font, state, foreground color, background color, surface to draw on, and the bounds to draw within. The **System.Drawing.Graphics** surface on which to draw. The **System.Drawing.Font** to use, usually the parent control's **System.Drawing.Font** property. The **System.Drawing.Rectangle** bounds to draw within. The **System.Windows.Forms.Control.ControlCollection** index value of the item that is being drawn. The control's **System.Windows.Forms.DrawItemState** information. The foreground **System.Drawing.Color** to draw the control with. The background **System.Drawing.Color** to draw the control with.

f) *BackColor*

g) *ToString*

|           |         |            |           |                      |
|-----------|---------|------------|-----------|----------------------|
| [C#]      | public  | Color      | BackColor | {get;}               |
| [C++]     | public: | __property | Color     | get_BackColor();     |
| [VB]      | Public  | ReadOnly   | Property  | BackColor As Color   |
| [JScript] | public  | function   | get       | BackColor() : Color; |

#### Description

Gets the background color of the item that is being drawn.

If the item's state is **System.Windows.Forms.DrawItemState.Selected**, the **System.Windows.Forms.DrawItemEventArgs.BackColor** is set to **System.Drawing.SystemColors.HighlightText**. If the item's state is not **System.Windows.Forms.DrawItemState.Selected**, then the **System.Windows.Forms.DrawItemEventArgs.BackColor** property is set to **System.Drawing.SystemColors.Window**.

h) *Bounds*

i) *ToString*

[C#] public Rectangle Bounds {get;}

[C++] public: \_\_property Rectangle get\_Bounds();

[VB] Public ReadOnly Property Bounds As Rectangle

[JScript] public function get Bounds() : Rectangle;

*Description*

Gets the rectangle that represents the bounds of the item that is being drawn.

j) *Font*

k) *ToString*

[C#] public Font Font {get;}

[C++] public: \_\_property Font\* get\_Font();

[VB] Public ReadOnly Property Font As Font

[JScript] public function get Font() : Font;

*Description*

Gets the font assigned to the item being drawn.

A suggested **System.Drawing.Font** , usually the parent control's **System.Windows.Forms.Control.Font** property.



l) *ForeColor*

m) *ToString*

```
[C#]          public          Color          ForeColor          {get;}
[C++]          public:          __property          Color          get_ForeColor();
[VB]          Public          ReadOnly          Property          ForeColor          As          Color
[JavaScript]    public          function          get          ForeColor()          :          Color;
```

*Description*

Gets the foreground color of the of the item being drawn.

If the item's state is **System.Windows.Forms.DrawItemState.Selected** , the **System.Windows.Forms.DrawItemEventArgs.ForeColor** is set to **System.Drawing.SystemColors.HighlightText** . If the item's state is not **System.Windows.Forms.DrawItemState.Selected** , then the **System.Windows.Forms.DrawItemEventArgs.ForeColor** property is set to **System.Drawing.SystemColors.WindowText** .

n) *Graphics*

o) *ToString*

```
[C#]          public          Graphics          Graphics          {get;}
[C++]          public:          __property          Graphics*          get_Graphics();
[VB]          Public          ReadOnly          Property          Graphics          As          Graphics
[JavaScript]    public          function          get          Graphics()          :          Graphics;
```

*Description*

Gets the graphics surface to draw the item on.

p) *Index*

q) *ToString*

[C#]            public            int            Index            {get;}

[C++]           public:           \_\_property           int           get\_Index();

[VB]        Public        ReadOnly        Property        Index        As        Integer

[JScript]        public        function        get        Index()        :        int;

*Description*

Gets the index value of the item that is being drawn.

This property returns the

**System.Windows.Forms.Control.ControlCollection.Item(System.Int32)**

value of the item being drawn in the

**System.Windows.Forms.Control.ControlCollection** .

r) *State*

s) *ToString*

[C#]            public            DrawItemState            State            {get;}

[C++]           public:           \_\_property           DrawItemState           get\_State();

[VB]        Public        ReadOnly        Property        State        As        DrawItemState

[JScript]        public        function        get        State()        :        DrawItemState;

*Description*

Gets the state of the item being drawn.

This property value may be a combination of the

**System.Windows.Forms.DrawItemState** enumeration members. The

members can be combined by use of bitwise operators.

t) *DrawBackground*

|           |             |          |      |                   |
|-----------|-------------|----------|------|-------------------|
| [C#]      | public      | virtual  | void | DrawBackground(); |
| [C++]     | public:     | virtual  | void | DrawBackground(); |
| [VB]      | Overridable | Public   | Sub  | DrawBackground()  |
| [JScript] | public      | function |      | DrawBackground(); |

*Description*

Draws the background within the bounds specified in the **System.Windows.Forms.DrawItemEventArgs.#ctor** constructor and with the appropriate color.

If the item being drawn is **System.Windows.Forms.DrawItemState.Selected**, the background is drawn with the text highlighted.

u) *DrawFocusRectangle*

|           |             |          |      |                       |
|-----------|-------------|----------|------|-----------------------|
| [C#]      | public      | virtual  | void | DrawFocusRectangle(); |
| [C++]     | public:     | virtual  | void | DrawFocusRectangle(); |
| [VB]      | Overridable | Public   | Sub  | DrawFocusRectangle()  |
| [JScript] | public      | function |      | DrawFocusRectangle(); |

*Description*

Draws a focus rectangle within the bounds specified in the **System.Windows.Forms.DrawItemEventArgs.#ctor** constructor.

If the item being drawn has **System.Windows.Forms.DrawItemState.Focus**, the focus rectangle is drawn.

DrawItemEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **DrawItem** event of a **System.Windows.Forms.ComboBox** , **System.Windows.Forms.ListBox** , **System.Windows.Forms.MenuItem** , or **System.Windows.Forms.TabControl** control. The source of the event. A **System.Windows.Forms.DrawItemEventArgs** that contains the event data.

When you create a **System.Windows.Forms.DrawItemEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

DrawItemState enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies the state of an item that is being drawn.

This enumeration is used by members such as **System.Windows.Forms.DrawItemEventArgs.State** .

b) *ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | Checked;         |
| [C++]     | public: | const | DrawItemState | Checked;         |
| [VB]      | Public  | Const | Checked       | As DrawItemState |
| [JScript] | public  | var   | Checked       | : DrawItemState; |

*Description*

The item is checked. Only menu controls use this value.

*c) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | ComboBoxEdit;    |
| [C++]     | public: | const | DrawItemState | ComboBoxEdit;    |
| [VB]      | Public  | Const | ComboBoxEdit  | As DrawItemState |
| [JScript] | public  | var   | ComboBoxEdit  | : DrawItemState; |

*Description*

The item is the editing portion of a **System.Windows.Forms.ComboBox** .

*d) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | Default;         |
| [C++]     | public: | const | DrawItemState | Default;         |
| [VB]      | Public  | Const | Default       | As DrawItemState |
| [JScript] | public  | var   | Default       | : DrawItemState; |

*Description*

The item is in its default visual state.

*e) ToString*

|       |         |       |               |           |
|-------|---------|-------|---------------|-----------|
| [C#]  | public  | const | DrawItemState | Disabled; |
| [C++] | public: | const | DrawItemState | Disabled; |

|           |        |       |          |    |                |
|-----------|--------|-------|----------|----|----------------|
| [VB]      | Public | Const | Disabled | As | DrawItemState  |
| [JScript] | public | var   | Disabled | :  | DrawItemState; |

*Description*

The item is disabled.

*f) ToString*

|      |        |       |               |        |
|------|--------|-------|---------------|--------|
| [C#] | public | const | DrawItemState | Focus; |
|------|--------|-------|---------------|--------|

|       |         |       |               |        |
|-------|---------|-------|---------------|--------|
| [C++] | public: | const | DrawItemState | Focus; |
|-------|---------|-------|---------------|--------|

|      |        |       |       |    |               |
|------|--------|-------|-------|----|---------------|
| [VB] | Public | Const | Focus | As | DrawItemState |
|------|--------|-------|-------|----|---------------|

|           |        |     |       |   |                |
|-----------|--------|-----|-------|---|----------------|
| [JScript] | public | var | Focus | : | DrawItemState; |
|-----------|--------|-----|-------|---|----------------|

*Description*

The item has focus.

*g) ToString*

|      |        |       |               |         |
|------|--------|-------|---------------|---------|
| [C#] | public | const | DrawItemState | Grayed; |
|------|--------|-------|---------------|---------|

|       |         |       |               |         |
|-------|---------|-------|---------------|---------|
| [C++] | public: | const | DrawItemState | Grayed; |
|-------|---------|-------|---------------|---------|

|      |        |       |        |    |               |
|------|--------|-------|--------|----|---------------|
| [VB] | Public | Const | Grayed | As | DrawItemState |
|------|--------|-------|--------|----|---------------|

|           |        |     |        |   |                |
|-----------|--------|-----|--------|---|----------------|
| [JScript] | public | var | Grayed | : | DrawItemState; |
|-----------|--------|-----|--------|---|----------------|

*Description*

The item is grayed. Only menu controls use this value.

***h) ToString***

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | HotLight;        |
| [C++]     | public: | const | DrawItemState | HotLight;        |
| [VB]      | Public  | Const | HotLight      | As DrawItemState |
| [JScript] | public  | var   | HotLight      | : DrawItemState; |

***Description***

The item is being hot-tracked (the item is highlighted as the mouse pointer passes over it).

***i) ToString***

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | Inactive;        |
| [C++]     | public: | const | DrawItemState | Inactive;        |
| [VB]      | Public  | Const | Inactive      | As DrawItemState |
| [JScript] | public  | var   | Inactive      | : DrawItemState; |

***Description***

The item is inactive.

***j) ToString***

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | NoAccelerator;   |
| [C++]     | public: | const | DrawItemState | NoAccelerator;   |
| [VB]      | Public  | Const | NoAccelerator | As DrawItemState |
| [JScript] | public  | var   | NoAccelerator | : DrawItemState; |

*Description*

The item displays without a keyboard accelerator.

*k) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | NoFocusRect;     |
| [C++]     | public: | const | DrawItemState | NoFocusRect;     |
| [VB]      | Public  | Const | NoFocusRect   | As DrawItemState |
| [JScript] | public  | var   | NoFocusRect   | : DrawItemState; |

*Description*

The item displays without the visual cue that indicates it has focus.

*l) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | DrawItemState | None;            |
| [C++]     | public: | const | DrawItemState | None;            |
| [VB]      | Public  | Const | None          | As DrawItemState |
| [JScript] | public  | var   | None          | : DrawItemState; |

*Description*

The item currently has no state.

*m) ToString*

|       |         |       |               |           |
|-------|---------|-------|---------------|-----------|
| [C#]  | public  | const | DrawItemState | Selected; |
| [C++] | public: | const | DrawItemState | Selected; |



|           |        |       |          |    |                |
|-----------|--------|-------|----------|----|----------------|
| [VB]      | Public | Const | Selected | As | DrawItemState  |
| [JScript] | public | var   | Selected | :  | DrawItemState; |

*Description*

The item is selected.

DrawMode enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies how the elements of a control are drawn.

This enumeration is used by members, such as **System.Windows.Forms.ListBox.DrawMode** in the **System.Windows.Forms.ListBox** , **System.Windows.Forms.CheckedListBox** , and **System.Windows.Forms.ComboBox** classes.

*b) ToString*

|           |         |       |          |             |
|-----------|---------|-------|----------|-------------|
| [C#]      | public  | const | DrawMode | Normal;     |
| [C++]     | public: | const | DrawMode | Normal;     |
| [VB]      | Public  | Const | Normal   | As DrawMode |
| [JScript] | public  | var   | Normal   | : DrawMode; |

*Description*

All the elements in a control are drawn by the operating system and are of the same size.

c) *ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | DrawMode       | OwnerDrawFixed; |
| [C++]     | public: | const | DrawMode       | OwnerDrawFixed; |
| [VB]      | Public  | Const | OwnerDrawFixed | As DrawMode     |
| [JScript] | public  | var   | OwnerDrawFixed | : DrawMode;     |

*Description*

All the elements in the control are drawn manually and are of the same size.

d) *ToString*

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C#]      | public  | const | DrawMode          | OwnerDrawVariable; |
| [C++]     | public: | const | DrawMode          | OwnerDrawVariable; |
| [VB]      | Public  | Const | OwnerDrawVariable | As DrawMode        |
| [JScript] | public  | var   | OwnerDrawVariable | : DrawMode;        |

*Description*

All the elements in the control are drawn manually and may differ in size.

ErrorBlinkStyle enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies constants indicating when the error icon, supplied by an **System.Windows.Forms.ErrorProvider**, should blink to alert the user that an error has occurred.

This enumeration is used by **System.Windows.Forms.ErrorProvider** .

*b) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | ErrorBlinkStyle | AlwaysBlink;       |
| [C++]     | public: | const | ErrorBlinkStyle | AlwaysBlink;       |
| [VB]      | Public  | Const | AlwaysBlink     | As ErrorBlinkStyle |
| [JScript] | public  | var   | AlwaysBlink     | : ErrorBlinkStyle; |

*Description*

Always blink when the error icon is first displayed, or when a error description string is set for the control and the error icon is already displayed.

*c) ToString*

|           |         |       |                       |                        |
|-----------|---------|-------|-----------------------|------------------------|
| [C#]      | public  | const | ErrorBlinkStyle       | BlinkIfDifferentError; |
| [C++]     | public: | const | ErrorBlinkStyle       | BlinkIfDifferentError; |
| [VB]      | Public  | Const | BlinkIfDifferentError | As ErrorBlinkStyle     |
| [JScript] | public  | var   | BlinkIfDifferentError | : ErrorBlinkStyle;     |

*Description*

Blinks when the icon is already displayed and a new error string is set for the control.

*d) ToString*

|       |         |       |                 |                    |
|-------|---------|-------|-----------------|--------------------|
| [C#]  | public  | const | ErrorBlinkStyle | NeverBlink;        |
| [C++] | public: | const | ErrorBlinkStyle | NeverBlink;        |
| [VB]  | Public  | Const | NeverBlink      | As ErrorBlinkStyle |

1 [JScript] public var NeverBlink : ErrorBlinkStyle;

2  
3 *Description*

4 Never blink the error icon.

5 ErrorIconAlignment enumeration (System.Windows.Forms)

6 *a) ToString*

7  
8  
9 *Description*

10 Specifies constants indicating the locations that an error icon can appear in  
11 relation to the control with an error.

12 This enumeration is used by **System.Windows.Forms.ErrorProvider** .

13 *b) ToString*

14 [C#] public const ErrorIconAlignment BottomLeft;

15 [C++] public: const ErrorIconAlignment BottomLeft;

16 [VB] Public Const BottomLeft As ErrorIconAlignment

17 [JScript] public var BottomLeft : ErrorIconAlignment;

18  
19 *Description*

20 The icon appears aligned with the bottom of the control and the left of the  
21 control.

22 *c) ToString*

23  
24 [C#] public const ErrorIconAlignment BottomRight;

```

1  [C++]      public:      const      ErrorIconAlignment      BottomRight;
2  [VB]      Public      Const      BottomRight      As      ErrorIconAlignment
3  [JScript]  public      var      BottomRight      :      ErrorIconAlignment;

```

#### *Description*

The icon appears aligned with the bottom of the control and the right of the control.

#### *d) ToString*

```

9  [C#]      public      const      ErrorIconAlignment      MiddleLeft;
10 [C++]      public:      const      ErrorIconAlignment      MiddleLeft;
11 [VB]      Public      Const      MiddleLeft      As      ErrorIconAlignment
12 [JScript]  public      var      MiddleLeft      :      ErrorIconAlignment;

```

#### *Description*

The icon appears aligned with the middle of the control and the left of the control.

#### *e) ToString*

```

19 [C#]      public      const      ErrorIconAlignment      MiddleRight;
20 [C++]      public:      const      ErrorIconAlignment      MiddleRight;
21 [VB]      Public      Const      MiddleRight      As      ErrorIconAlignment
22 [JScript]  public      var      MiddleRight      :      ErrorIconAlignment;

```

#### *Description*

The icon appears aligned with the middle of the control and the right of the control.

*f) ToString*

|           |         |       |                    |                       |
|-----------|---------|-------|--------------------|-----------------------|
| [C#]      | public  | const | ErrorIconAlignment | TopLeft;              |
| [C++]     | public: | const | ErrorIconAlignment | TopLeft;              |
| [VB]      | Public  | Const | TopLeft            | As ErrorIconAlignment |
| [JScript] | public  | var   | TopLeft            | : ErrorIconAlignment; |

*Description*

The icon appears aligned with the top of the control and to the left of the control.

*g) ToString*

|           |         |       |                    |                       |
|-----------|---------|-------|--------------------|-----------------------|
| [C#]      | public  | const | ErrorIconAlignment | TopRight;             |
| [C++]     | public: | const | ErrorIconAlignment | TopRight;             |
| [VB]      | Public  | Const | TopRight           | As ErrorIconAlignment |
| [JScript] | public  | var   | TopRight           | : ErrorIconAlignment; |

*Description*

The icon appears aligned with the top of the control and to the right of the control.

ErrorProvider class (System.Windows.Forms)

a) *ToString*

### *Description*

Provides a user interface for indicating that a control on a form has an error associated with it.

**System.Windows.Forms.ErrorProvider** presents a simple interface for indicating to the end user that a control on a form has an error associated with it. If an error description string is specified for the control, an icon appears next to the control. The icon flashes in the manner specified by **System.Windows.Forms.ErrorProvider.BlinkStyle**, at the rate specified by **System.Windows.Forms.ErrorProvider.BlinkRate**. When the mouse hovers over the icon, a tool tip appears showing the error description string.

b) *ErrorProvider*

*Example Syntax:*

c) *ToString*

|           |                 |                  |
|-----------|-----------------|------------------|
| [C#]      | public          | ErrorProvider(); |
| [C++]     | public:         | ErrorProvider(); |
| [VB]      | Public          | Sub New()        |
| [JScript] | public function | ErrorProvider(); |

Initializes a new instance of the **System.Windows.Forms.ErrorProvider** class.

### *Description*

Initializes a new instance of the **System.Windows.Forms.ErrorProvider** class and initializes the default settings for **System.Windows.Forms.ErrorProvider.BlinkRate**, **System.Windows.Forms.ErrorProvider.BlinkStyle**, and the **System.Windows.Forms.ErrorProvider.Icon**.

The following table shows initial property values for an instance of **System.Windows.Forms.ErrorProvider** .

*d) ErrorProvider*

*Example Syntax:*

*e) ToString*

```
[C#]      public      ErrorProvider(ContainerControl      parentControl);
[C++]      public:      ErrorProvider(ContainerControl*      parentControl);
[VB]      Public      Sub      New(ByVal      parentControl      As      ContainerControl)
[JScript]      public      function      ErrorProvider(parentControl      :      ContainerControl);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ErrorProvider** class attached to a container.

The **System.Windows.Forms.Control** specified by **System.Windows.Forms.ErrorProvider.ContainerControl** is the container object for the data-bound controls to associate the error provider with. The container of the control to monitor for errors.

*f) BlinkRate*

*g) ToString*

```
[C#]      public      int      BlinkRate      {get;      set;}
[C++]      public:      __property      int      get_BlinkRate();public:      __property      void
set_BlinkRate(int);
[VB]      Public      Property      BlinkRate      As      Integer
[JScript]      public      function      get BlinkRate() : int;public      function      set BlinkRate(int);
```



*Description*

Gets or sets the rate at which the error icon flashes.

The error icon flashes at the specified rate. A value of zero sets **System.Windows.Forms.ErrorProvider.BlinkStyle** to **System.Windows.Forms.ErrorBlinkStyle.NeverBlink** .

*h) BlinkStyle*

*i) ToString*

[C#] public ErrorBlinkStyle BlinkStyle {get; set;}

[C++] public: \_\_property ErrorBlinkStyle get\_BlinkStyle();public: \_\_property void set\_BlinkStyle(ErrorBlinkStyle);

[VB] Public Property BlinkStyle As ErrorBlinkStyle

[JScript] public function get BlinkStyle() : ErrorBlinkStyle;public function set BlinkStyle(ErrorBlinkStyle);

*Description*

Gets or sets a value indicating when the error icon flashes.

The error icon flashes in the manner specified by the assigned **System.Windows.Forms.ErrorBlinkStyle** when an error occurs. Setting the **System.Windows.Forms.ErrorProvider.BlinkRate** to zero sets the **System.Windows.Forms.ErrorProvider.BlinkStyle** to **System.Windows.Forms.ErrorBlinkStyle.NeverBlink** .

j) *Container*

k) *ContainerControl*

l) *ToString*

*Description*

Gets or sets a value indicating the parent control for this **System.Windows.Forms.ErrorProvider** .

Typically, this is the **System.Windows.Forms.Form** the data-bound controls reside on.

m) *DataMember*

n) *ToString*

[C#]            public            string            DataMember            {get;            set;}

[C++] public: \_\_property String\* get\_DataMember();public: \_\_property void  
set\_DataMember(String\*);

[VB]            Public            Property            DataMember            As            String

[JScript] public function get DataMember() : String;public function set  
DataMember(String);

*Description*

Gets or sets the data table to monitor.

The **System.Windows.Forms.ErrorProvider.DataMember** is a navigation string based on **System.Windows.Forms.ErrorProvider.DataSource** .

o) *DataSource*

p) *ToString*

[C#]        public        object        DataSource        {get;        set;}

[C++] public: \_\_property Object\* get\_DataSource();public: \_\_property void  
set\_DataSource(Object\*);

[VB]        Public        Property        DataSource        As        Object

[JScript] public function get DataSource() : Object;public function set  
DataSource(Object);

#### *Description*

Gets or sets data set the **System.Windows.Forms.ErrorProvider** to monitor.

The **System.Windows.Forms.ErrorProvider.DataSource** is a  
**System.Data.DataSet** containing tables with the fields that you can attach to a  
control and that you want to monitor for errors.

q) *DesignMode*

r) *Events*

s) *Icon*

t) *ToString*

#### *Description*

Gets or sets the **System.Drawing.Icon** that is displayed next to a control when  
an error description string has been set for the control.

For best results, using an icon of the size 16 by 16 pixels. If the specified icon is  
not 16 by 16, it is resized to 16 by 16.

u) *Site*

v) *ToString*

[C#]                      ISite                      Site                      {set;}

[C++]      public:      \_\_property      virtual      void      set\_Site(ISite\*);

[VB]                      Property                      Site                      As                      ISite

[JScript]                      public                      function                      set                      Site(ISite);

*Description*

w) *BindToDataAndErrors*

[C#]      public      void      BindToDataAndErrors(object      newDataSource,      string  
newDataMember);

[C++]      public:      void      BindToDataAndErrors(Object\*      newDataSource,      String\*  
newDataMember);

[VB] Public Sub BindToDataAndErrors(ByVal newDataSource As Object, ByVal  
newDataMember                      As                      String)

[JScript]      public      function      BindToDataAndErrors(newDataSource      :      Object,  
newDataMember                      :                      String);

*Description*

Provides a method to set both the  
**System.Windows.Forms.ErrorProvider.DataSource** and  
**System.Windows.Forms.ErrorProvider.DataMember** at run time.

To avoid conflicts at run time that can occur when changing  
**System.Windows.Forms.ErrorProvider.DataSource** and

**System.Windows.Forms.ErrorProvider.DataMember** , you should use **System.Windows.Forms.ErrorProvider.BindToDataAndErrors(System.Object, System.String)** instead of setting **System.Windows.Forms.ErrorProvider.DataSource** and **System.Windows.Forms.ErrorProvider.DataMember** individually. A **System.Data.DataSet** to monitor for errors. A collection within the **System.Data.DataSet** to monitor for errors.

#### x) *CanExtend*

```
[C#]      public      bool      CanExtend(object      extendee);
[C++]     public:      __sealed    bool      CanExtend(Object*      extendee);
[VB]      NotOverridable Public Function CanExtend(ByVal extendee As Object) As
Boolean
[JScript] public  function  CanExtend(extendee : Object) : Boolean;
```

#### *Description*

Gets a value indicating whether a control can be extended.

*Return Value:* **true** if the control can be extended; otherwise, **false** .

Typically, you will use

**System.Windows.Forms.ErrorProvider.CanExtend(System.Object)** to determine whether you can attach an **System.Windows.Forms.ErrorProvider** to the specified control. The control to be extended.

#### y) *Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]     protected:      void      Dispose(bool      disposing);
[VB]      Overrides Protected Sub Dispose(ByVal disposing As Boolean)
[JScript] protected override function Dispose(disposing : Boolean);
```

## Description

Disposes of the resources (other than memory) used by the **System.Windows.Forms.ErrorProvider**.

Call **System.Windows.Forms.ErrorProvider.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.ErrorProvider**. The **System.Windows.Forms.ErrorProvider.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.ErrorProvider** in an unusable state. After calling **System.Windows.Forms.ErrorProvider.Dispose(System.Boolean)**, you must release all references to the **System.Windows.Forms.ErrorProvider** so the memory it was occupying can be reclaimed by garbage collection.

### z) *GetError*

```
[C#]      public      string      GetError(Control      control);
[C++]     public:     String*      GetError(Control*      control);
[VB]      Public Function GetError(ByVal control As Control) As String
[JScript] public function GetError(control : Control) : String;
```

## Description

Returns the current error description string for the specified control.

*Return Value:* The error description string for the specified control. The item to get the error description string for.

### aa) *GetIconAlignment*

```
[C#]      public      ErrorIconAlignment GetIconAlignment(Control      control);
[C++]     public:     ErrorIconAlignment GetIconAlignment(Control*      control);
[VB]      Public Function GetIconAlignment(ByVal control As Control) As
ErrorIconAlignment
```

1 [JScript] public function GetIconAlignment(control : Control) :  
2 ErrorIconAlignment;

3  
4 *Description*

5 Gets a value indicating where the error icon should be placed in relation to the  
6 control.

7 *Return Value:* One of the **System.Windows.Forms.ErrorIconAlignment**  
8 values. The control to get the icon location for.

9  
10 *bb) GetIconPadding*

11 [C#] public int GetIconPadding(Control control);

12 [C++] public: int GetIconPadding(Control\* control);

13 [VB] Public Function GetIconPadding(ByVal control As Control) As Integer

14 [JScript] public function GetIconPadding(control : Control) : int;

15  
16 *Description*

17 Returns the amount of extra space to leave next to the error icon.

18 *Return Value:* The number of pixels to leave between the icon and the control.  
19 Many icons normally have extra space around their central images, so the  
20 padding value is only necessary if additional space is necessary. Padding values  
21 can be positive or negative. Negative values cause the icon to overlap the edge  
22 of the control. The control to get the padding for.

23  
24 *cc) SetError*

25 [C#] public void SetError(Control control, string value);

[C++] public: void SetError(Control\* control, String\* value);

[VB] Public Sub SetError(ByVal control As Control, ByVal value As String)

[JScript] public function SetError(control : Control, value : String);

## *Description*

Sets the error description string for the specified control.

If the string length is greater than zero, then the error icon is displayed, and the ToolTip for the error icon is the error description text. If the string length is zero, the error icon is hidden. The control to set the error description string for. The error description string.

### *dd) SetIconAlignment*

```
[C#] public void SetIconAlignment(Control control, ErrorIconAlignment value);
```

```
[C++] public: void SetIconAlignment(Control* control, ErrorIconAlignment value);
```

```
[VB] Public Sub SetIconAlignment(ByVal control As Control, ByVal value As ErrorIconAlignment)
```

```
[JScript] public function SetIconAlignment(control : Control, value : ErrorIconAlignment);
```

## *Description*

Sets the location where the error icon should be placed in relation to the control.

The final placement of the icon is modified by the icon padding values. The control to set the icon location for. One of the **System.Windows.Forms.ErrorIconAlignment** values.

### *ee) SetIconPadding*

```
[C#] public void SetIconPadding(Control control, int padding);
```

```
[C++] public: void SetIconPadding(Control* control, int padding);
```

```
[VB] Public Sub SetIconPadding(ByVal control As Control, ByVal padding As
```



Integer)

[JScript] public function SetIconPadding(control : Control, padding : int);

#### *Description*

Sets the amount of extra space to leave between the specified control and the error icon.

Many icons normally have extra space around their central images, so the padding value is only necessary when additional space is necessary. Padding values can be positive or negative. Negative values cause the icon to overlap the edge of the control. The *control* to set the padding for. The number of pixels to add between the icon and the *control*.

#### *ff) UpdateBinding*

|      |        |      |                  |
|------|--------|------|------------------|
| [C#] | public | void | UpdateBinding(); |
|------|--------|------|------------------|

|       |         |      |                  |
|-------|---------|------|------------------|
| [C++] | public: | void | UpdateBinding(); |
|-------|---------|------|------------------|

|      |        |     |                 |
|------|--------|-----|-----------------|
| [VB] | Public | Sub | UpdateBinding() |
|------|--------|-----|-----------------|

|           |        |          |                  |
|-----------|--------|----------|------------------|
| [JScript] | public | function | UpdateBinding(); |
|-----------|--------|----------|------------------|

#### *Description*

Provides a method to update the bindings of the **System.Windows.Forms.ErrorProvider.DataSource**, **System.Windows.Forms.ErrorProvider.DataMember**, and the error text.

Typically, you call this method after you have called **System.Windows.Forms.ErrorProvider.BindToDataAndErrors(System.Object, System.String)**.

FeatureSupport class (System.Windows.Forms)

*a) UpdateBinding*

*Description*

Provides **static** methods for retrieving feature information from the current system.

Use the **static** methods of this class when the classes you query for feature information implement the **System.Windows.Forms.IFeatureSupport** interface. Otherwise, inherit from **System.Windows.Forms.FeatureSupport** and provide your own implementation. For an implementation of this class, see **System.Windows.Forms.OSFeature** .

*b) FeatureSupport*

*Example Syntax:*

*c) UpdateBinding*

|           |                    |                   |
|-----------|--------------------|-------------------|
| [C#]      | protected          | FeatureSupport(); |
| [C++]     | protected:         | FeatureSupport(); |
| [VB]      | Protected          | Sub New()         |
| [JScript] | protected function | FeatureSupport(); |

*d) GetVersionPresent*

|           |              |          |          |                                     |                    |
|-----------|--------------|----------|----------|-------------------------------------|--------------------|
| [C#]      | public       | abstract | Version  | GetVersionPresent(object            | feature);          |
| [C++]     | public:      | virtual  | Version* | GetVersionPresent(Object*           | feature) = 0;      |
| [VB]      | MustOverride | Public   | Function | GetVersionPresent(ByVal             | feature As Object) |
| As        |              |          |          |                                     | Version            |
| [JScript] | public       | abstract | function | GetVersionPresent(feature : Object) | : Version;         |

## Description

When overridden in a derived class, gets the version of the specified feature that is available on the system.

**Return Value:** A **System.Version** representing the version number of the specified feature available on the system; or **null** if the feature is not installed.

Version numbers consist of three parts: major, minor, and build. Typically, a version number is displayed as "major number.minor number.build number". The feature whose version is requested.

### e) *GetVersionPresent*

```
[C#] public static Version GetVersionPresent(string featureClassName, string  
featureConstName);
```

```
[C++] public: static Version* GetVersionPresent(String* featureClassName,  
String* featureConstName);
```

```
[VB] Public Shared Function GetVersionPresent(ByVal featureClassName As  
String, ByVal featureConstName As String) As Version
```

```
[JScript] public static function GetVersionPresent(featureClassName : String,  
featureConstName : String) : Version; Gets the version of the specified feature that  
is available on the system.
```

## Description

Gets the version of the specified feature that is available on the system.

**Return Value:** A **System.Version** with the version number of the specified feature available on the system; or **null** if the feature is not installed.

Version numbers consist of three parts: major, minor, and build. Typically, a version number is displayed as "major number.minor number.build number". The fully qualified name of the class to query for information about the specified feature. This class must implement the

**System.Windows.Forms.IFeatureSupport** interface or inherit from a class that implements this interface. The fully qualified name of the feature to look for.

*f) IsPresent*

[C#] public virtual bool IsPresent(object feature);

[C++] public: virtual bool IsPresent(Object\* feature);

[VB] Overridable Public Function IsPresent(ByVal feature As Object) As Boolean

[JScript] public function IsPresent(feature : Object) : Boolean;

*Description*

Determines whether any version of the specified feature is installed in the system.

*Return Value:* **true** if the feature is present; otherwise, **false** .

When you inherit from **System.Windows.Forms.FeatureSupport** , you must override the

**System.Windows.Forms.FeatureSupport.GetVersionPresent(System.String, System.String)** method. When you override this method, check that the class that you use for the *feature* parameter is the same as the class used for this parameter in the

**System.Windows.Forms.FeatureSupport.IsPresent(System.String, System.String)** method. If the two *feature* parameters differ, you must also override **System.Windows.Forms.FeatureSupport.IsPresent(System.String, System.String)** . The feature to look for.

*g) IsPresent*

[C#] public virtual bool IsPresent(object feature, Version minimumVersion);

[C++] public: virtual bool IsPresent(Object\* feature, Version\* minimumVersion);

[VB] Overridable Public Function IsPresent(ByVal feature As Object, ByVal minimumVersion As Version) As Boolean

[JScript] public function IsPresent(feature : Object, minimumVersion : Version) :

Boolean;

### *Description*

Determines whether the specified or newer version of the specified feature is installed in the system.

*Return Value:* **true** if the feature is present and its version number is greater than or equal to the specified minimum version number; **false** if the feature is not installed or its version number is below the specified minimum number.

When you inherit from **System.Windows.Forms.FeatureSupport**, you must override the

**System.Windows.Forms.FeatureSupport.GetVersionPresent(System.String, System.String)** method. When you override this method, check that the class that you use for the *feature* parameter is the same as the class used for this parameter in the

**System.Windows.Forms.FeatureSupport.IsPresent(System.String, System.String)** method. If the two *feature* parameters differ, you must also override **System.Windows.Forms.FeatureSupport.IsPresent(System.String, System.String)**. The feature to look for. A **System.Version** representing the minimum version number of the feature to look for.

#### *h) IsPresent*

```
[C#] public static bool IsPresent(string featureClassName, string  
featureConstName);
```

```
[C++] public: static bool IsPresent(String* featureClassName, String*  
featureConstName);
```

```
[VB] Public Shared Function IsPresent(ByVal featureClassName As String,  
ByVal featureConstName As String) As Boolean
```

```
[JScript] public static function IsPresent(featureClassName : String,  
featureConstName : String) : Boolean; Determines whether the specified feature is  
installed in the system.
```

## Description

Determines whether any version of the specified feature is installed in the system. This method is **static**.

**Return Value:** **true** if the specified feature is present; **false** if the specified feature is not present or if the product containing the feature is not installed.

See the documentation for the product containing the feature to determine the names to pass to the *featureClassName* and the *featureConstName* parameters. The fully qualified name of the class to query for information about the specified feature. This class must implement the **System.Windows.Forms.IFeatureSupport** interface or inherit from a class that implements this interface. The fully qualified name of the feature to look for.

### i) *IsPresent*

```
[C#] public static bool IsPresent(string featureClassName, string  
featureConstName, Version minimumVersion);
```

```
[C++] public: static bool IsPresent(String* featureClassName, String*  
featureConstName, Version* minimumVersion);
```

```
[VB] Public Shared Function IsPresent(ByVal featureClassName As String,  
ByVal featureConstName As String, ByVal minimumVersion As Version) As  
Boolean
```

```
[JScript] public static function IsPresent(featureClassName : String,  
featureConstName : String, minimumVersion : Version) : Boolean;
```

## Description

Determines whether the specified or newer version of the specified feature is installed in the system. This method is **static**.

**Return Value:** **true** if the feature is present and its version number is greater than or equal to the specified minimum version number; **false** if the feature is not installed or its version number is below the specified minimum number.

See the documentation for the product containing the feature to determine the names to pass to the *featureClassName* and the *featureConstName* parameters. The fully qualified name of the class to query for information about the specified feature. This class must implement the **System.Windows.Forms.IFeatureSupport** interface or inherit from a class that implements this interface. The fully qualified name of the feature to look for. A **System.Version** representing the minimum version number of the feature.

FileDialog class (System.Windows.Forms)

a) *ToString*

#### *Description*

Displays a dialog window from which the user can select a file.

**System.Windows.Forms.FileDialog** is an abstract class, and cannot be instantiated directly. Additionally, you cannot inherit from this class. To create a dialog to select or save a file use **System.Windows.Forms.OpenFileDialog** or **System.Windows.Forms.SaveFileDialog**.

b) *ToString*

|           |            |        |          |              |              |
|-----------|------------|--------|----------|--------------|--------------|
| [C#]      | protected  | static | readonly | object       | EventFileOk; |
| [C++]     | protected: | static | Object*  | EventFileOk; |              |
| [VB]      | Protected  | Shared | ReadOnly | EventFileOk  | As Object    |
| [JScript] | protected  | static | var      | EventFileOk  | : Object;    |

#### *Description*

1           c)     *AddExtension*

2           d)     *ToString*

3  
4 [C#]       public       bool       AddExtension       {get;       set;}

5 [C++] public: \_\_property bool get\_AddExtension();public: \_\_property void  
6 set\_AddExtension(bool);

7 [VB]       Public       Property       AddExtension       As       Boolean

8 [JScript] public function get AddExtension() : Boolean;public function set  
9 AddExtension(Boolean);

10  
11 *Description*

12 Gets or sets a value indicating whether the dialog box automatically adds an  
13 extension to a file name if the user omits the extension.

14 The extension added to a file name depends on the currently selected file filter  
15 and the value of the **System.Windows.Forms.FileDialog.CheckFileExists**  
16 property.

17           e)     *CheckFileExists*

18           f)     *ToString*

19 [C#]       public       virtual       bool       CheckFileExists       {get;       set;}

20 [C++] public: \_\_property virtual bool get\_CheckFileExists();public: \_\_property  
21 virtual                               void                               set\_CheckFileExists(bool);

22 [VB]       Overridable   Public       Property       CheckFileExists       As       Boolean

23 [JScript] public function get CheckFileExists() : Boolean;public function set  
24 CheckFileExists(Boolean);



1  
2 *Description*

3 Gets or sets a value indicating whether the dialog box displays a warning if the  
4 user specifies a file name that does not exist.

5 The default value is **true** for an inheriting  
6 **System.Windows.Forms.OpenFileDialog** and **false** for an inheriting  
7 **System.Windows.Forms.SaveFileDialog** .

8 *g) CheckPathExists*

9 *h) ToString*

10 [C#] public bool CheckPathExists {get; set;}

11 [C++] public: \_\_property bool get\_CheckPathExists();public: \_\_property void  
12 set\_CheckPathExists(bool);

13 [VB] Public Property CheckPathExists As Boolean

14 [JScript] public function get CheckPathExists() : Boolean;public function set  
15 CheckPathExists(Boolean);

16  
17 *Description*

18 Gets or sets a value indicating whether the dialog box displays a warning if the  
19 user specifies a path that does not exist.

20 *i) Container*

21 *j) DefaultExt*

22 *k) ToString*

23  
24 *Description*  
25

Gets or sets the default file extension.

*l) DereferenceLinks*

*m) ToString*

[C#] public bool DereferenceLinks {get; set;}

[C++] public: \_\_property bool get\_DereferenceLinks();public: \_\_property void set\_DereferenceLinks(bool);

[VB] Public Property DereferenceLinks As Boolean

[JScript] public function get DereferenceLinks() : Boolean;public function set DereferenceLinks(Boolean);

#### *Description*

Gets or sets a value indicating whether the dialog box returns the location of the file referenced by the shortcut or whether it returns the location of the shortcut (.lnk).

*n) DesignMode*

*o) Events*

*p) FileName*

*q) ToString*

#### *Description*

Gets or sets a string containing the file name selected in the file dialog box.

The file name includes both the file path and the extension. If no files are selected, this method returns an empty string ("").

1            **r)      *FileNames***

2            **s)      *ToString***

3  
4    [C#]            public            string[]            FileNames            {get;}

5    [C++]            public:            \_\_property            String\*            get\_FileNames();

6    [VB]    Public    ReadOnly    Property    FileNames    As    String    ()

7    [JScript]    public    function    get    FileNames()    :    String[];

8  
9    *Description*

10   Gets the file names of all selected files in the dialog box.

11   Each file name includes both the file path and the extension. If no files are  
12   selected, this method returns an empty array.

13            **t)      *Filter***

14            **u)      *ToString***

15    [C#]            public            string            Filter            {get;            set;}

16    [C++]    public:    \_\_property    String\*    get\_Filter();public:    \_\_property    void  
17    set\_Filter(String\*);

18    [VB]            Public            Property            Filter            As            String

19    [JScript] public function get Filter() : String;public function set Filter(String);

20  
21    *Description*

22   Gets or sets the current file name filter string, which determines the choices that  
23   appear in the "Save as file type" or "Files of type" box in the dialog box.

24   For each filtering option, the filter string contains a description of the filter,  
25   followed by the vertical bar (|) and the filter pattern. The strings for different  
filtering options are separated by the vertical bar.

v) *FilterIndex*

w) *ToString*

```
[C#]      public      int      FilterIndex      {get;      set;}
[C++]    public:  __property  int  get_FilterIndex();public:  __property  void
set_FilterIndex(int);
[VB]      Public      Property      FilterIndex      As      Integer
[JScript] public function get FilterIndex() : int;public function set FilterIndex(int);
```

#### *Description*

Gets or sets the index of the filter currently selected in the file dialog box.

x) *InitialDirectory*

y) *ToString*

```
[C#]      public      string      InitialDirectory      {get;      set;}
[C++]    public:  __property  String*  get_InitialDirectory();public:  __property  void
set_InitialDirectory(String*);
[VB]      Public      Property      InitialDirectory      As      String
[JScript] public function get InitialDirectory() : String;public function set
InitialDirectory(String);
```

#### *Description*

Gets or sets the initial directory displayed by the file dialog box.

z) *Instance*

aa) *ToString*

```
[C#]      protected      virtual      IntPtr      Instance      {get;}
[C++]      protected:      __property      virtual      IntPtr      get_Instance();
[VB]      Overridable      Protected      ReadOnly      Property      Instance      As      IntPtr
[JScript]      protected      function      get      Instance()      :      IntPtr;
```

*Description*

Gets the Win32 instance handle for the application.

bb) *Options*

cc) *ToString*

```
[C#]      protected      int      Options      {get;}
[C++]      protected:      __property      int      get_Options();
[VB]      Protected      ReadOnly      Property      Options      As      Integer
[JScript]      protected      function      get      Options()      :      int;
```

*Description*

Gets the Win32 common Open File Dialog OFN\_\* option flags.

dd) *RestoreDirectory*

ee) *ToString*

```
[C#]      public      bool      RestoreDirectory      {get;      set;}
[C++]      public:      __property      bool      get_RestoreDirectory();public:      __property      void
```

1 set\_RestoreDirectory(bool);

2 [VB] Public Property RestoreDirectory As Boolean

3 [JScript] public function get RestoreDirectory() : Boolean;public function set

4 RestoreDirectory(Boolean);

5  
6 *Description*

7 Gets or sets a value indicating whether the dialog box restores the current  
8 directory before closing.

9 *ff) ShowHelp*

10 *gg) ToString*

11  
12 [C#] public bool ShowHelp {get; set;}

13 [C++] public: \_\_property bool get\_ShowHelp();public: \_\_property void  
14 set\_ShowHelp(bool);

15 [VB] Public Property ShowHelp As Boolean

16 [JScript] public function get ShowHelp() : Boolean;public function set

17 ShowHelp(Boolean);

18  
19 *Description*

20 Gets or sets a value indicating whether the **Help** button is displayed in the file  
21 dialog.

22 A **System.Windows.Forms.Control.HelpRequested** event is raised when  
23 the user clicks the **Help** button.  
24  
25

*hh) Site*

*ii) Title*

*jj) ToString*

*Description*

Gets or sets the file dialog box title.

The string is placed in the title bar of the dialog box. If the title is an empty string, the system uses a default title, which is either "Save As" or "Open".

*kk) ValidateNames*

*ll) ToString*

[C#]        public        bool        ValidateNames        {get;        set;}

[C++] public: \_\_property bool get\_ValidateNames();public: \_\_property void  
set\_ValidateNames(bool);

[VB]        Public        Property        ValidateNames        As        Boolean

[JScript] public function get ValidateNames() : Boolean;public function set  
ValidateNames(Boolean);

*Description*

Gets or sets a value indicating whether the dialog box accepts only valid Win32 file names.

If the edit control contains anything but spaces when the user clicks **OK**, the dialog returns the file name, whether it is valid or not. No default extension is added to the text.

*mm) ToString*

*Description*

Occurs when the user clicks on the Open or Save button on a file dialog box.

For information about handling events, see .

For information about handling events, see .

*nn) HookProc*

[C#] protected override IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam, IntPtr lParam);

[C++] protected: IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam, IntPtr lParam);

[VB] Overrides Protected Function HookProc(ByVal hWnd As IntPtr, ByVal msg As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As IntPtr

[JScript] protected override function HookProc(hWnd : IntPtr, msg : int, wParam : IntPtr, lParam : IntPtr) : IntPtr;

*Description*

Defines the common dialog box hook procedure that is overridden to add specific functionality to the file dialog box.

*Return Value:* Returns zero if the default dialog box procedure processes the message; returns a nonzero value if the default dialog box procedure ignores the message.

A hook procedure allows the user to connect or insert other routines into a routine or application for the purpose of debugging or enhancing functionality. The handle to the dialog box window. The message received by the dialog box. Additional information about the message. Additional information about the message.



*oo) OnFileOk*

```
[C#]      protected      void      OnFileOk(CancelEventArgs      e);
[C++]      protected:      void      OnFileOk(CancelEventArgs*      e);
[VB]      Protected      Sub      OnFileOk(ByVal      e      As      CancelEventArgs)
[JScript]      protected      function      OnFileOk(e      :      CancelEventArgs);
```

*Description*

Raises the **System.Windows.Forms.FileDialog.FileOk** event. A **System.ComponentModel.CancelEventArgs** that contains the event data.

*pp) Reset*

```
[C#]      public      override      void      Reset();
[C++]      public:      void      Reset();
[VB]      Overrides      Public      Sub      Reset()
[JScript]      public      override      function      Reset();
```

*Description*

Resets all properties to their default values.

When overriding **System.Windows.Forms.FileDialog.Reset** in a derived class, be sure to call the base class's **System.Windows.Forms.CommonDialog.Reset** method.

*qq) RunDialog*

```
[C#]      protected      override      bool      RunDialog(IntPtr      hWndOwner);
[C++]      protected:      bool      RunDialog(IntPtr      hWndOwner);
```

[VB] Overrides Protected Function RunDialog(ByVal hWndOwner As IntPtr) As Boolean

[JScript] protected override function RunDialog(hWndOwner : IntPtr) : Boolean;

#### *Description*

Specifies a common dialog box.

*Return Value:* **true** if the file could be opened; otherwise, **false** .

This method provides an implementation of

**System.Windows.Forms.CommonDialog.RunDialog(System.IntPtr)** ,

and is invoked when the user of a file dialog invokes

**System.Windows.Forms.CommonDialog.ShowDialog** . A value that represents the window handle of the owner window for the common dialog box.

#### *rr) ToString*

[C#]            public            override            string            ToString();

[C++]            public:            String\*            ToString();

[VB]    Overrides    Public    Function    ToString()    As    String

[JScript]    public    override    function    ToString()    :    String;

#### *Description*

Provides a string version of this Object.

*Return Value:* A string version of this Object.

FlatStyle enumeration (System.Windows.Forms)

#### *a) ToString*

#### *Description*

Specifies the appearance of a control.

This enumeration is used by members such as **System.Windows.Forms.ButtonBase.FlatStyle** , **System.Windows.Forms.GroupBox.FlatStyle** , and **System.Windows.Forms.Label.FlatStyle** .

*b) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | FlatStyle | Flat;        |
| [C++]     | public: | const | FlatStyle | Flat;        |
| [VB]      | Public  | Const | Flat      | As FlatStyle |
| [JScript] | public  | var   | Flat      | : FlatStyle; |

*Description*

The control appears flat.

*c) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | FlatStyle | Popup;       |
| [C++]     | public: | const | FlatStyle | Popup;       |
| [VB]      | Public  | Const | Popup     | As FlatStyle |
| [JScript] | public  | var   | Popup     | : FlatStyle; |

*Description*

A control appears flat until the mouse pointer moves over it, at which point it appears three-dimensional.

*d) ToString*

|       |         |       |           |           |
|-------|---------|-------|-----------|-----------|
| [C#]  | public  | const | FlatStyle | Standard; |
| [C++] | public: | const | FlatStyle | Standard; |

|           |        |       |          |    |            |
|-----------|--------|-------|----------|----|------------|
| [VB]      | Public | Const | Standard | As | FlatStyle  |
| [JScript] | public | var   | Standard | :  | FlatStyle; |

*Description*

The control appears three-dimensional.

*e) ToString*

|           |         |       |           |         |            |
|-----------|---------|-------|-----------|---------|------------|
| [C#]      | public  | const | FlatStyle | System; |            |
| [C++]     | public: | const | FlatStyle | System; |            |
| [VB]      | Public  | Const | System    | As      | FlatStyle  |
| [JScript] | public  | var   | System    | :       | FlatStyle; |

*Description*

The appearance of the control is determined by the user's operating system.

FontDialog class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a common dialog box that displays a list of fonts that are currently installed on the system.

The inherited member

**System.Windows.Forms.CommonDialog.ShowDialog** must be invoked to create this specific common dialog box.

**b) ToString**

|           |            |        |          |             |             |
|-----------|------------|--------|----------|-------------|-------------|
| [C#]      | protected  | static | readonly | object      | EventApply; |
| [C++]     | protected: | static | Object*  | EventApply; |             |
| [VB]      | Protected  | Shared | ReadOnly | EventApply  | As Object   |
| [JScript] | protected  | static | var      | EventApply  | : Object;   |

*Description*

**c) FontDialog**

*Example Syntax:*

**d) ToString**

|           |         |                        |
|-----------|---------|------------------------|
| [C#]      | public  | FontDialog();          |
| [C++]     | public: | FontDialog();          |
| [VB]      | Public  | Sub New()              |
| [JScript] | public  | function FontDialog(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.FontDialog** class.

When you create an instance of **System.Windows.Forms.FontDialog** , the following read/write properties are initialized.

e) *AllowScriptChange*

f) *ToString*

[C#] public bool AllowScriptChange {get; set;}

[C++] public: \_\_property bool get\_AllowScriptChange();public: \_\_property void set\_AllowScriptChange(bool);

[VB] Public Property AllowScriptChange As Boolean

[JScript] public function get AllowScriptChange() : Boolean;public function set AllowScriptChange(Boolean);

#### *Description*

Gets or sets a value indicating whether the user can change the character set specified in the **Script** combo box to display a character set other than the one currently displayed.

The **Script** combo box found on the font dialog box contains character sets associated with the selected font.

g) *AllowSimulations*

h) *ToString*

[C#] public bool AllowSimulations {get; set;}

[C++] public: \_\_property bool get\_AllowSimulations();public: \_\_property void set\_AllowSimulations(bool);

[VB] Public Property AllowSimulations As Boolean

[JScript] public function get AllowSimulations() : Boolean;public function set AllowSimulations(Boolean);

*Description*

Gets or sets a value indicating whether the dialog box allows graphics device interface (GDI) font simulations.

*i) AllowVectorFonts*

*j) ToString*

[C#] public bool AllowVectorFonts {get; set;}

[C++] public: \_\_property bool get\_AllowVectorFonts();public: \_\_property void set\_AllowVectorFonts(bool);

[VB] Public Property AllowVectorFonts As Boolean

[JScript] public function get AllowVectorFonts() : Boolean;public function set AllowVectorFonts(Boolean);

*Description*

Gets or sets a value indicating whether the dialog box allows vector font selections.

*k) AllowVerticalFonts*

*l) ToString*

[C#] public bool AllowVerticalFonts {get; set;}

[C++] public: \_\_property bool get\_AllowVerticalFonts();public: \_\_property void set\_AllowVerticalFonts(bool);

[VB] Public Property AllowVerticalFonts As Boolean

[JScript] public function get AllowVerticalFonts() : Boolean;public function set

1 AllowVerticalFonts(Boolean);

3 *Description*

4 Gets or sets a value indicating whether the dialog box displays both vertical and  
horizontal fonts or only horizontal fonts.

5 *m) Color*

7 *n) ToString*

8 [C#] public Color Color {get; set;}

9 [C++] public: \_\_property Color get\_Color();public: \_\_property void  
10 set\_Color(Color);

11 [VB] Public Property Color As Color

12 [JScript] public function get Color() : Color;public function set Color(Color);

14 *Description*

15 Gets or sets the selected font color.

16 *o) Container*

17 *p) DesignMode*

18 *q) Events*

19 *r) FixedPitchOnly*

20 *s) ToString*

22 *Description*



1 Gets or sets a value indicating whether the dialog box allows only the selection  
of fixed-pitch fonts.

2        *t)        Font*

3        *u)        ToString*

4  
5 [C#]        public        Font        Font        {get;        set;}

6 [C++]    public:    \_\_property    Font\*    get\_Font();public:    \_\_property    void  
7 set\_Font(Font\*);

8 [VB]        Public        Property        Font        As        Font

9 [JScript] public function get Font() : Font;public function set Font(Font);

10  
11 *Description*

12 Gets or sets the selected font.

13        *v)        FontMustExist*

14        *w)        ToString*

15  
16 [C#]        public        bool        FontMustExist        {get;        set;}

17 [C++]    public:    \_\_property    bool    get\_FontMustExist();public:    \_\_property    void  
18 set\_FontMustExist(bool);

19 [VB]        Public        Property        FontMustExist        As        Boolean

20 [JScript] public function get FontMustExist() : Boolean;public function set  
21 FontMustExist(Boolean);

22  
23 *Description*

24 Gets or sets a value indicating whether the dialog box specifies an error  
25 condition if the user attempts to select a font or style that does not exist.

x) *MaxSize*

y) *ToString*

[C#]            public            int            MaxSize            {get;            set;}

[C++]   public:   \_\_property   int   get\_MaxSize();public:   \_\_property   void  
set\_MaxSize(int);

[VB]            Public            Property            MaxSize            As            Integer

[JScript] public function get MaxSize() : int;public function set MaxSize(int);

#### *Description*

Gets or sets the maximum point size a user can select.

In order for the maximum and minimum size settings to take effect,  
**System.Windows.Forms.FontDialog.MaxSize** must be greater than  
**System.Windows.Forms.FontDialog.MinSize** , and both must be greater  
than zero.

z) *MinSize*

aa) *ToString*

[C#]            public            int            MinSize            {get;            set;}

[C++]   public:   \_\_property   int   get\_MinSize();public:   \_\_property   void  
set\_MinSize(int);

[VB]            Public            Property            MinSize            As            Integer

[JScript] public function get MinSize() : int;public function set MinSize(int);

#### *Description*

Gets or sets the minimum point size a user can select.

In order for the maximum and minimum size settings to take effect, **System.Windows.Forms.FontDialog.MaxSize** must be greater than **System.Windows.Forms.FontDialog.MinSize**, and both must be greater than zero.

*bb) Options*

*cc) ToString*

```
[C#]          protected          int          Options          {get;}
[C++]          protected:          __property          int          get_Options();
[VB]          Protected          ReadOnly          Property          Options          As          Integer
[JScript]          protected          function          get          Options()          :          int;
```

#### *Description*

Gets the value passed to CHOOSEFONT.Flags.

*dd) ScriptsOnly*

*ee) ToString*

```
[C#]          public          bool          ScriptsOnly          {get;          set;}
[C++]          public:          __property          bool          get_ScriptsOnly();public:          __property          void
set_ScriptsOnly(bool);
[VB]          Public          Property          ScriptsOnly          As          Boolean
[JScript]          public          function          get          ScriptsOnly()          :          Boolean;public          function          set
ScriptsOnly(Boolean);
```

#### *Description*

Gets or sets a value indicating whether the dialog box allows selection of fonts for all non-OEM and Symbol character sets, as well as the ANSI character set.

*ff) ShowApply*

*gg) ToString*

[C#]            public            bool            ShowApply            {get;            set;}

[C++] public: \_\_property bool get\_ShowApply();public: \_\_property void  
set\_ShowApply(bool);

[VB]            Public            Property            ShowApply            As            Boolean

[JScript] public function get ShowApply() : Boolean;public function set  
ShowApply(Boolean);

*Description*

Gets or sets a value indicating whether the dialog box contains an Apply button.

*hh) ShowColor*

*ii) ToString*

[C#]            public            bool            ShowColor            {get;            set;}

[C++] public: \_\_property bool get\_ShowColor();public: \_\_property void  
set\_ShowColor(bool);

[VB]            Public            Property            ShowColor            As            Boolean

[JScript] public function get ShowColor() : Boolean;public function set  
ShowColor(Boolean);

*Description*

Gets or sets a value indicating whether the dialog box displays the color choice.

1        *jj) ShowEffects*

2        *kk) ToString*

3  
4 [C#]        public        bool        ShowEffects        {get;        set;}

5 [C++] public: \_\_property bool get\_ShowEffects();public: \_\_property void  
6 set\_ShowEffects(bool);

7 [VB]        Public        Property        ShowEffects        As        Boolean

8 [JScript] public function get ShowEffects() : Boolean;public function set  
9 ShowEffects(Boolean);

10  
11 *Description*

12 Gets or sets a value indicating whether the dialog box contains controls that  
13 allow the user to specify strikethrough, underline, and text color options.

14        *ll) ShowHelp*

15        *mm) ToString*

16 [C#]        public        bool        ShowHelp        {get;        set;}

17 [C++] public: \_\_property bool get\_ShowHelp();public: \_\_property void  
18 set\_ShowHelp(bool);

19 [VB]        Public        Property        ShowHelp        As        Boolean

20 [JScript] public function get ShowHelp() : Boolean;public function set  
21 ShowHelp(Boolean);

22  
23 *Description*

24 Gets or sets a value indicating whether the dialog box displays a Help button.  
25

*nn) Site*

*oo) ToString*

### *Description*

Occurs when the user clicks the Apply button in the font dialog box.

Every time the Apply button is clicked, another **System.Windows.Forms.FontDialog.Apply** event is raised.

*pp) HookProc*

[C#] protected override IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam, IntPtr lParam);

[C++] protected: IntPtr HookProc(IntPtr hWnd, int msg, IntPtr wParam, IntPtr lParam);

[VB] Overrides Protected Function HookProc(ByVal hWnd As IntPtr, ByVal msg As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As IntPtr

[JScript] protected override function HookProc(hWnd : IntPtr, msg : int, wParam : IntPtr, lParam : IntPtr) : IntPtr;

### *Description*

Specifies the common dialog box hook procedure that is overridden to add specific functionality to a common dialog box.

**Return Value:** A zero value if the default dialog box procedure processes the message; a nonzero value if the default dialog box procedure ignores the message.

Raising an event invokes the event handler through a delegate. For an overview, see . The handle to the dialog box window. The message being received. Additional information about the message. Additional information about the message.

## qq) OnApply

```
[C#]      protected      virtual      void      OnApply(EventArgs      e);
[C++]     protected:     virtual      void      OnApply(EventArgs*      e);
[VB]      Overridable     Protected     Sub      OnApply(ByVal e As EventArgs)
[JScript]      protected      function      OnApply(e      :      EventArgs);
```

### Description

Raises the **System.Windows.Forms.FontDialog.Apply** event.

Raising an event invokes the event handler through a delegate. For an overview, see . An **System.EventArgs** that contains the data.

## rr) Reset

```
[C#]      public      override      void      Reset();
[C++]     public:      void      Reset();
[VB]      Overrides     Public      Sub      Reset()
[JScript]      public      override      function      Reset();
```

### Description

Resets all dialog box options to their default values.

When the options are reset, the strikethrough, underline, and color effects are enabled, the fonts listed include only the screen fonts supported by the system.

## ss) RunDialog

```
[C#]      protected      override      bool      RunDialog(IntPtr      hWndOwner);
[C++]     protected:      bool      RunDialog(IntPtr      hWndOwner);
```

1 [VB] Overrides Protected Function RunDialog(ByVal hWndOwner As IntPtr) As

2 Boolean

3 [JScript] protected override function RunDialog(hWndOwner : IntPtr) : Boolean;

4  
5 *Description*

6 The actual implementation of running the dialog. Inheriting classes should  
7 override this if they want to add more functionality, and call base.runDialog() if  
8 necessary

*Return Value:* **true** if the user hit OK; otherwise **false** . **UNDONE**

8  
9 *tt) ToString*

10 [C#] public override string ToString();

11 [C++] public: String\* ToString();

12 [VB] Overrides Public Function ToString() As String

13 [JScript] public override function ToString() : String;

14  
15  
16 *Description*

17 Retrieves a string that includes the name of the current font selected in the  
18 dialog box.

*Return Value:* A string that includes the name of the currently selected font.

19 Form class (System.Windows.Forms)

20 *a) ToString*

21  
22  
23 *Description*

24 Represents a window or dialog box that makes up an application's user interface.

25 A **System.Windows.Forms.Form** is a representation of any window displayed  
in your application. The **System.Windows.Forms.Form** class can be used to



create standard, tool, borderless, and floating windows. You can also use the **System.Windows.Forms.Form** class to create modal windows such as a dialog box. A special kind of form, the multiple document interface (MDI) form, can contain other forms called MDI child forms. An MDI form is created by setting the **System.Windows.Forms.Form.IsMdiContainer** property to **true**. MDI child forms are created by setting the **System.Windows.Forms.Form.MdiParent** property to the MDI parent form that will contain the child form.

*b) Form*

*Example Syntax:*

*c) ToString*

|           |                 |         |
|-----------|-----------------|---------|
| [C#]      | public          | Form(); |
| [C++]     | public:         | Form(); |
| [VB]      | Public Sub      | New()   |
| [JScript] | public function | Form(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.Form** class.

The default size of a form is 300 pixels in height and 300 pixels in width.

*d) AcceptButton*

*e) ToString*

|           |                    |                    |                                    |                     |
|-----------|--------------------|--------------------|------------------------------------|---------------------|
| [C#]      | public             | IButtonControl     | AcceptButton                       | {get; set;}         |
| [C++]     | public: __property | IButtonControl*    | get_AcceptButton();                | public: __property  |
|           | void               |                    | set_AcceptButton(IButtonControl*); |                     |
| [VB]      | Public             | Property           | AcceptButton                       | As IButtonControl   |
| [JScript] | public function    | get AcceptButton() | : IButtonControl;                  | public function set |

1 AcceptButton(IconButtonControl);

2  
3 *Description*

4 Gets or sets the button on the form that is clicked when the user presses the  
5 ENTER key.

6 This property allows you to designate a default action to occur when the user  
7 presses the ENTER key in your application. The button assigned to this property  
8 must be an **System.Windows.Forms.IconButtonControl** that is on the current  
9 form or located within a container on the current form.

- 10 *f) AccessibilityObject*
- 11 *g) AccessibleDefaultActionDescription*
- 12 *h) AccessibleDescription*
- 13 *i) AccessibleName*
- 14 *j) AccessibleRole*
- 15 *k) ActiveControl*
- 16 *l) ActiveForm*
- 17 *m) ToString*

18  
19 *Description*

20 Gets the currently active form for this application.

21 You can use this method to obtain a reference to the currently active form to  
22 perform actions on the form or its controls.

23

24

25

n) *ActiveMdiChild*

o) *ToString*

```
[C#]      public      Form      ActiveMdiChild      {get;}
[C++]     public:      __property      Form*      get_ActiveMdiChild();
[VB]      Public      ReadOnly      Property      ActiveMdiChild      As      Form
[JScript] public      function      get      ActiveMdiChild()      :      Form;
```

#### *Description*

Gets the currently active multiple document interface (MDI) child window.

You can use this method to determine whether there are any MDI child forms open in your MDI application. You can also use this method to perform operations on an MDI child window from its MDI parent form or from another form that is displayed in your application.

p) *AllowDrop*

q) *AllowTransparency*

r) *ToString*

#### *Description*

Gets or sets a value indicating whether the opacity of the form can be adjusted.

This property is automatically set to **true** if the Opacity is changed. When the opacity of a form is set using the Opacity property, the form will layer objects on the form. If you set the **System.Windows.Forms.Form.AllowTransparency** property to **false** the form will not be in layered mode which will improve the display performance of the form.

1       s)     *Anchor*

2       t)     *AutoScale*

3       u)     *ToString*

4  
5  
6     *Description*

7     Gets or sets a value indicating whether the form adjusts its size to fit the height  
8     of the font used on the form and scales its controls.

9     You can use this property to allow your form and its controls to automatically  
10    adjust based on changes in the font. This can be useful in applications where the  
11    font might increase or decrease based on the language specified for use by  
12    Windows.

13  
14       v)     *AutoScaleBaseSize*

15       w)     *ToString*

16 [C#]     public     virtual     Size     AutoScaleBaseSize     {get;     set;}  
17

18 [C++] public: \_\_property virtual Size get \_AutoScaleBaseSize();public: \_\_property  
19 virtual                               void                               set \_AutoScaleBaseSize(Size);  
20

21 [VB]     Overridable     Public     Property     AutoScaleBaseSize     As     Size  
22

23 [JScript] public function get AutoScaleBaseSize() : Size;public function set  
24 AutoScaleBaseSize(Size);  
25

26  
27     *Description*

28     Gets or sets the base size used for autoscaling of the form.

29     The value of the **System.Windows.Forms.Form.AutoScaleBaseSize**  
30     property is used at form-display time to compute the scaling factor for the form.  
31     The autoscaling base size is used by the form as a baseline for comparison to the  
32     system's font size to determine how much to scale the form when autoscaling is  
33     used. If you want to determine the size a form will auto scale to based on a

specific font, use the

**System.Windows.Forms.Form.GetAutoScaleSize(System.Drawing.Font)**  
method.

x) *AutoScroll*

y) *ToString*

[C#] public override bool AutoScroll {get; set;}

[C++] public: \_\_property virtual bool get\_AutoScroll();public: \_\_property virtual  
void set\_AutoScroll(bool);

[VB] Overrides Public Property AutoScroll As Boolean

[JScript] public function get AutoScroll() : Boolean;public function set  
AutoScroll(Boolean);

### *Description*

Gets or sets a value indicating whether the form implements autoscrolling.

If this property is set to **true**, scroll bars are displayed on the form if any controls are located outside the form's client region. Additionally, when autoscrolling is on, the client area of the form automatically scrolls to make the control with input focus visible.

z) *AutoScrollMargin*

aa) *AutoScrollMinSize*

bb) *AutoScrollPosition*

cc) *BackColor*

dd) *ToString*

### *Description*

The background color of this control. This is an ambient property and will always return a non-null value.

*ee) BackgroundImage*

*ff) BindingContext*

*gg) Bottom*

*hh) Bounds*

*ii) CancelButton*

*jj) ToString*

#### *Description*

Gets or sets the button control that is clicked when the user presses the ESC key.

The cancel button for a form is the button control that is clicked whenever the user presses the ESC key. The button assigned to this property must be an **System.Windows.Forms.IButtonControl** that is on the current form or located within a container on the current form.

1        *kk) CanFocus*

2        *ll) CanSelect*

3        *mm) Capture*

4        *nn) CausesValidation*

5        *oo) ClientRectangle*

6        *pp) ClientSize*

7        *qq) ToString*

8  
9  
10      *Description*

11      Gets or sets the size of the client area of the form.

12      The size of the client area of the form is the size of the form excluding the  
13      borders and the title bar. The client area of a form is the area within a form  
14      where controls can be placed. You can use this property to get the proper  
15      dimensions when performing graphics operations or when sizing and positioning  
16      controls on the form. To get the size of the entire form, use the  
17      **System.Windows.Forms.Form.Size** property or use the individual properties  
18      **System.Windows.Forms.Control.Height** and  
19      **System.Windows.Forms.Control.Width** .  
20  
21  
22  
23  
24  
25

1        *rr)    **CompanyName***

2        *ss)    **Container***

3        *tt)    **ContainsFocus***

4        *uu)    **ContextMenu***

5        *vv)    **ControlBox***

6        *ww)    **ToString***

7  
8  
9        *Description*

10       Gets or sets a value indicating whether a control box is displayed in the caption  
11       bar of the form.

12       If the **System.Windows.Forms.Form.ControlBox** property is set to **true** ,  
13       the control box is displayed in the upper-left corner of the caption bar. The  
14       control box is where the user can click to access the system menu.

15       *xx)    **Controls***

16       *yy)    **Created***

17       *zz)    **CreateParams***

18       *aaa)   **ToString***

19  
20       *Description*

21       Retrieves the CreateParams used to create the window. If a subclass overrides  
22       this function, it must call the base implementation.



*bbb) Cursor*

*ccc) DataBindings*

*ddd) DefaultImeMode*

*eee) ToString*

*Description*

*fff) DefaultSize*

*ggg) ToString*

[C#]       protected       override       Size       DefaultSize       {get;}

[C++]     protected:     \_\_property     virtual     Size     get\_DefaultSize();

[VB]   Overrides   Protected   ReadOnly   Property   DefaultSize   As   Size

[JScript]     protected     function     get     DefaultSize()     :     Size;

*Description*

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.

*hhh) DesignMode*

*iii) DesktopBounds*

*jjj) ToString*

*Description*

Gets or sets the size and location of the form on the Windows desktop.

Desktop coordinates are based on the working area of the screen, which excludes the taskbar. The coordinate system of the desktop is pixel based. If your application is running on a multiple monitor system, the coordinates of the form are the coordinates for the combined desktop.

**kkk) DesktopLocation**

**lll) ToString**

[C#]        public        Point        DesktopLocation        {get;        set;}

[C++] public: \_\_property Point get\_DesktopLocation();public: \_\_property void  
set\_DesktopLocation(Point);

[VB]        Public        Property        DesktopLocation        As        Point

[JScript] public function get DesktopLocation() : Point;public function set  
DesktopLocation(Point);

#### *Description*

Gets or sets the location of the form on the Windows desktop.

Desktop coordinates are based on the working area of the screen, which excludes the taskbar. The coordinate system of the desktop is pixel based. If your application is running on a multi-monitor system, the coordinates of the form are the coordinates for the combined desktop.

**mmm) DialogResult**

**nnn) ToString**

[C#]        public        DialogResult        DialogResult        {get;        set;}

[C++] public: \_\_property DialogResult get\_DialogResult();public: \_\_property  
void        set\_DialogResult(DialogResult);

[VB]      Public      Property      DialogResult      As      DialogResult  
[JScript] public function get DialogResult() : DialogResult; public function set  
DialogResult(DialogResult);

*Description*

Gets or sets the dialog result for the form.

The dialog result of a form is the value that is returned from the form when it is displayed as a modal dialog. If the form is displayed as a dialog box, setting this property with a value from the **System.Windows.Forms.DialogResult** enumeration sets the value of the dialog result for the form, hides the modal dialog, and returns control to the calling form. This property is typically set by the **System.Windows.Forms.Button.DialogResult** property of a **System.Windows.Forms.Button** control on the form. When the user clicks the **System.Windows.Forms.Button** control, the value assigned to the **System.Windows.Forms.Button.DialogResult** property of the **System.Windows.Forms.Button** is assigned to the **System.Windows.Forms.Form.DialogResult** property of the form.

ooo) *DisplayRectangle*

ppp) *Disposing*

qqq) *Dock*

rrr) *DockPadding*

sss) *Enabled*

ttt) *Events*

uuu) *Focused*

vvv) *Font*

www) *FontHeight*

xxx) *ForeColor*

yyy) *FormBorderStyle*

zzz) *ToString*

### *Description*

Gets or sets the border style of the form.

The border style of the form determines how the outer edge of the form appears. In addition to changing the border display for a form, certain border styles prevent the form from being sized. For example, the **FormBorderStyle.FixedDialog** border style changes the border of the form to that of a dialog box and prevents the form from being resized. The border style can also affect the size or availability of the caption bar section of a form.

### *Description*

Gets or sets the border style of the form.

1        *aaaa) Handle*

2        *bbbb) HasChildren*

3        *cccc) Height*

4        *dddd) HelpButton*

5        *eeee) ToString*

6  
7  
8        *Description*

9        Gets or sets a value indicating whether a Help button should be displayed in the  
10       caption box of the form.

11       When this property is set to **true** , a small button with a question mark appears  
12       in the caption bar to the left of the close button. You can use this button to  
13       display help for your application. You can create an event handler for the  
14       **System.Windows.Forms.Control.HelpRequested** event of the  
15       **System.Windows.Forms.Control** class to display Help information to the user  
16       when the Help button of the form is clicked.

17        *ffff) HScroll*

18        *gggg) Icon*

19        *hhhh) ToString*

20       *Description*

21       Gets or sets the icon for the form.

22       A form's icon designates the picture that represents the form in the taskbar as  
23       well as the icon that is displayed for the control box of the form.

1        *iiii) ImeMode*

2        *jjjj) InvokeRequired*

3        *kkkk) IsAccessible*

4        *llll) IsDisposed*

5        *mmmm)IsHandleCreated*

6        *nnnn) IsMdiChild*

7        *oooo) ToString*

10        *Description*

11        Gets a value indicating whether the form is a multiple document interface (MDI)  
12        child form.

13        At run time, MDI child forms are displayed inside the client area of an MDI  
14        parent form. An MDI child form can be maximized, minimized, and moved within  
15        the MDI parent form. To create an MDI child form, assign the  
16        **System.Windows.Forms.Form** that will be the MDI parent form to the  
17        **System.Windows.Forms.Form.MdiParent** property of the child form. You  
18        can use the **System.Windows.Forms.Form.IsMdiContainer** property to  
19        determine whether a form is an MDI parent form.

17        *pppp) IsMdiContainer*

18        *qqqq) ToString*

20        [C#]        public        bool        IsMdiContainer        {get;        set;}

21        [C++]        public:        \_\_property bool get\_IsMdiContainer();public:        \_\_property void  
22        set\_IsMdiContainer(bool);

23        [VB]        Public        Property        IsMdiContainer        As        Boolean

24        [JScript]        public function get IsMdiContainer() : Boolean;public function set

IsMdiContainer(Boolean);

### Description

Gets or sets a value indicating whether the form is a container for multiple document interface (MDI) child forms.

This property changes the display and behavior of the form to an MDI parent form. When this property is set to **true**, the form displays a sunken client area with a raised border. All MDI child forms assigned to the parent form are displayed within its client area.

*rrrr) IsRestrictedWindow*

*ssss) ToString*

[C#]            public            bool            IsRestrictedWindow            {get;}

[C++]        public:        \_\_property        bool        get\_IsRestrictedWindow();

[VB]    Public    ReadOnly    Property    IsRestrictedWindow    As    Boolean

[JScript]    public    function    get    IsRestrictedWindow()    :    Boolean;

### Description

Determines if this form should display a warning banner when the form is displayed in an unsecure mode.

*tttt) KeyPreview*

*uuuu) ToString*

[C#]            public            bool            KeyPreview            {get;            set;}

[C++]    public:    \_\_property    bool    get\_KeyPreview();public:    \_\_property    void  
set\_KeyPreview(bool);

[VB]            Public            Property            KeyPreview            As            Boolean

[JScript] public function get KeyPreview() : Boolean; public function set KeyPreview(Boolean);

#### *Description*

Gets or sets a value indicating whether the form will receive key events before the event is passed to the control that has focus.

When this property is set to **true**, the form will receive all **System.Windows.Forms.Control.KeyPress**, **System.Windows.Forms.Control.KeyDown**, and **System.Windows.Forms.Control.KeyUp** events. After the form's event handlers have completed processing the keystroke, the keystroke is then assigned to the control with focus. For example, if the **System.Windows.Forms.Form.KeyPreview** property is set to **true** and the currently selected control is a **System.Windows.Forms.TextBox**, after the keystroke is handled by the event-handling methods of the form the **System.Windows.Forms.TextBox** control will receive the key that was pressed. To handle keyboard events only at the form level and not allow controls to receive keyboard events, set the *e.Handled* parameter in your form's **System.Windows.Forms.Control.KeyPress** or **System.Windows.Forms.Control.KeyDown** event to **true**.

vvvv) *Left*

www) *Location*

xxxx) *MaximizeBox*

yyyy) *ToString*

#### *Description*

Gets or sets a value indicating whether the maximize button is displayed in the caption bar of the form.

A maximize button enables users to enlarge a window to full-screen size. To display a maximize button, you must also set the form's **System.Windows.Forms.Form.FormBorderStyle** property to either



**FormBorderStyle.FixedSingle , FormBorderStyle.Sizable ,  
FormBorderStyle.Fixed3D , or FormBorderStyle.FixedDialog .**

*zzzz) MaximizedBounds*

*aaaaa) ToString*

[C#]       protected       Rectangle       MaximizedBounds       {get;       set;}

[C++]   protected:   \_\_property   Rectangle   get\_MaximizedBounds();protected:

\_\_property               void               set\_MaximizedBounds(Rectangle);

[VB]       Protected       Property       MaximizedBounds       As       Rectangle

[JScript]   protected   function   get   MaximizedBounds() : Rectangle;protected

function               set               MaximizedBounds(Rectangle);

#### *Description*

Gets the size of the form when it is maximized.

Classes that inherit from **System.Windows.Forms.Form** can override this method to provide new bounds for the form when it is maximized. The class sets this property internally when the form's maximize box button is clicked.

*bbbbbb) MaximumSize*

*ccccc) ToString*

[C#]       public       Size       MaximumSize       {get;       set;}

[C++]   public:   \_\_property   Size   get\_MaximumSize();public:   \_\_property   void

set\_MaximumSize(Size);

[VB]       Public       Property       MaximumSize       As       Size

[JScript]   public   function   get   MaximumSize() : Size;public   function   set

MaximumSize(Size);

## Description

Gets the maximum size the form can be resized to.

This property enables you to limit the size of a form to a specified maximum size. You can use this feature when displaying multiple windows at the same time, to ensure that a single window does not cause other windows to be hidden.

*dddd) MdiChildren*

*eeee) ToString*

```
[C#]          public          Form[]          MdiChildren          {get;}
[C++]          public:          __property          Form*          get_MdiChildren();
[VB]          Public          ReadOnly          Property          MdiChildren          As          Form          ()
[JavaScript]    public          function          get          MdiChildren()          :          Form[];
```

## Description

Gets an array of forms that represent the multiple document interface (MDI) child forms that are parented to this form.

This property allows you to obtain references to all the MDI child forms currently opened in an MDI parent form. To create an MDI child form, assign the **System.Windows.Forms.Form** that will be the MDI parent form to the **System.Windows.Forms.Form.MdiParent** property of the child form.

*ffff) MdiParent*

*gggg) ToString*

```
[C#]          public          Form          MdiParent          {get;          set;}
[C++]          public:          __property          Form*          get_MdiParent();public:          __property          void
set_MdiParent(Form*);
```

```

1 [VB]          Public          Property          MdiParent          As          Form
2 [JScript] public function get MdiParent() : Form;public function set
3 MdiParent(Form);

```

#### *Description*

Gets or sets the current multiple document interface (MDI) parent form of this form.

To create an MDI child form, assign the **System.Windows.Forms.Form** that will be the MDI parent form to the **System.Windows.Forms.Form.MdiParent** property of the child form. You can use this property from an MDI child form to obtain global information that all child forms need or to invoke methods that perform actions to all child forms.

*hhhhh)Menu*

*iiii) ToString*

```

14 [C#]          public          MainMenu          Menu          {get;          set;}
15 [C++] public: __property MainMenu* get_Menu();public: __property void
16 set_Menu(MainMenu*);
17 [VB]          Public          Property          Menu          As          MainMenu
18 [JScript] public function get Menu() : MainMenu;public function set
19 Menu(MainMenu);

```

#### *Description*

Gets or sets the **System.Windows.Forms.MainMenu** that is displayed in the form.

You can use this property to switch between complete menu sets at run time. For example, you can define one **System.Windows.Forms.MainMenu** to be displayed when your multiple document interface (MDI) form has no active MDI child forms and another **System.Windows.Forms.MainMenu** to display when

a child window is displayed. You can also use a different **System.Windows.Forms.MainMenu** when specific conditions exist in your application that require displaying a different menu set.

*jjjjj) MergedMenu*

*kkkkk) ToString*

```
[C#]      public      MainMenu      MergedMenu      {get;}
[C++]     public:     __property     MainMenu*       get_MergedMenu();
[VB]      Public     ReadOnly     Property     MergedMenu     As     MainMenu
[JScript] public     function     get     MergedMenu() :     MainMenu;
```

### *Description*

Gets the merged menu for the form.

This property is primarily used when the form is a multiple document interface (MDI) child form that merges its menu with its parent form's menu. You can use this property to obtain the current menu structure in an MDI application to make changes or additions to the menu structure. To obtain the non-merged **System.Windows.Forms.MainMenu** assigned to a form, use the **System.Windows.Forms.Form.Menu** property.

*lllll) MinimizeBox*

*mmmmm) ToString*

```
[C#]      public      bool      MinimizeBox      {get;      set;}
[C++]     public:     __property     bool     get_MinimizeBox();public:     __property     void
set_MinimizeBox(bool);
[VB]      Public     Property     MinimizeBox     As     Boolean
[JScript] public     function     get     MinimizeBox() : Boolean;public     function     set
MinimizeBox(Boolean);
```

*Description*

Gets or sets a value indicating whether the minimize button is displayed in the caption bar of the form.

A minimize button enables users to minimize a window to an icon. To display a minimize button, you must also set the form's

**System.Windows.Forms.Form.FormBorderStyle** property to either **FormBorderStyle.FixedSingle** , **FormBorderStyle.Sizable** , **FormBorderStyle.Fixed3D** , or **FormBorderStyle.FixedDialog** .

*nnnnn)MinimumSize*

*ooooo) ToString*

[C#]            public            Size            MinimumSize            {get;            set;}

[C++]   public:   \_\_property   Size   get\_MinimumSize();public:   \_\_property   void  
set\_MinimumSize(Size);

[VB]            Public            Property            MinimumSize            As            Size

[JScript]   public   function   get   MinimumSize()   :   Size;public   function   set  
MinimumSize(Size);

*Description*

Gets the minimum size the form can be resized to.

This property enables you to limit the size of a form to a specified minimum size. You can use this feature to prevent a user from sizing a window to an undesirable size.

*ppppp) Modal*

*qqqqq) ToString*

[C#]            public            bool            Modal            {get;}

|           |         |            |          |                    |
|-----------|---------|------------|----------|--------------------|
| [C++]     | public: | __property | bool     | get_Modal();       |
| [VB]      | Public  | ReadOnly   | Property | Modal As Boolean   |
| [JScript] | public  | function   | get      | Modal() : Boolean; |

#### Description

Gets a value indicating whether this form is displayed modally.

When a form is displayed modally, no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (usually in response to some user action) before input to another form can occur. Forms that are displayed modally are typically used as dialog boxes in an application.

*rrrrr) Name*

*sssss) Opacity*

*ttttt) ToString*

#### Description

Gets or sets the opacity level of the form.

This property enables you to specify a level of transparency for the form and its controls. This property differs from transparency provided by the **System.Windows.Forms.Form.TransparencyKey** which only makes a form and its controls completely transparent if they are the same color as the value specified in the **System.Windows.Forms.Form.TransparencyKey** property. When this property is set to a value less than 100% (1.00), the entire form, including borders, is made more transparent. Setting this property to a value of 0% (0.00) makes the form completely invisible. You can use this property to provide different levels of transparency or to provide affects such as phasing a form in or out of view. For example, you can phase a form into view by setting the **System.Windows.Forms.Form.Opacity** property to a value of 0% (0.00) and gradually increasing the value until it reaches 100% (1.00).

uuuuu)OwnedForms

vvvvv) ToString

```
[C#]      public      Form[]      OwnedForms      {get;}
[C++]     public:     __property  Form*      get_OwnedForms();
[VB]      Public      ReadOnly  Property  OwnedForms  As      Form      ()
[JScript] public      function  get      OwnedForms()      :      Form[];
```

#### Description

Gets an array of **System.Windows.Forms.Form** objects that represent all forms that are owned by this form.

This property returns an array that contains all forms that are owned by this form. To make a form owned by another form, call the **System.Windows.Forms.Form.AddOwnedForm(System.Windows.Forms.Form)** method. The form assigned to the owner form will remain owned until the **System.Windows.Forms.Form.RemoveOwnedForm(System.Windows.Forms.Form)** method is called. You can also make a form owned by another by setting the **System.Windows.Forms.Form.Owner** property with a reference to its owner form.

wwwww)Owner

xxxxx) ToString

```
[C#]      public      Form      Owner      {get;      set;}
[C++]     public:     __property  Form*      get_Owner();public:     __property  void
set_Owner(Form*);
[VB]      Public      Property      Owner      As      Form
[JScript] public      function  get Owner() : Form;public function set Owner(Form);
```

*Description*

Gets or sets the form that owns this form.

To make a form owned by another form, assign its **System.Windows.Forms.Form.Owner** property a reference to the form that will be the owner.

*yyyyy) Parent*

*zzzzz) ParentForm*

*aaaaaa)ProductName*

*bbbbbb)ProductVersion*

*cccccc)RecreatingHandle*

*dddddd)Region*

*eeeeee)RenderRightToLeft*

*ffffff) ResizeRedraw*

*gggggg)Right*

*hhhhhh)RightToLeft*

*iiiii) ShowFocusCues*

*jjjjj) ShowInTaskbar*

*kkkkkk)ToString*

*Description*

Gets or sets a value indicating whether the form is displayed in the Windows taskbar.



If a form is parented within another form, the parented form is not displayed in the Windows taskbar.

*llllll) ShowKeyboardCues*

*mmmmmm)Site*

*nnnnnn)Size*

*oooooo)ToString*

### *Description*

Gets or sets the size of the form.

This property allows you to set both the height and width of the form at the same time instead of setting the **System.Windows.Forms.Control.Height** and **System.Windows.Forms.Control.Width** properties individually. If you want to set the size and location of a form, you can use the **System.Windows.Forms.Form.DesktopBounds** property to size and locate the form based on desktop coordinates or use the **System.Windows.Forms.Control.Bounds** property of the **System.Windows.Forms.Control** class to set the size and location of the form based on screen coordinates.

*pppppp)SizeGripStyle*

*qqqqqq)ToString*

```
[C#]      public      SizeGripStyle      SizeGripStyle      {get;      set;}
```

```
[C++] public: __property SizeGripStyle get_SizeGripStyle();public: __property  
void      set_SizeGripStyle(SizeGripStyle);
```

```
[VB]      Public      Property      SizeGripStyle      As      SizeGripStyle
```

```
[JScript] public function get SizeGripStyle() : SizeGripStyle;public function set  
SizeGripStyle(SizeGripStyle);
```

## MS1-861US.APP

*ttttt) TabIndex*

*uuuuuu)ToString*

```
[C#]      public      new      int      TabIndex      {get;      set;}
[C++]    public:  __property  int  get_TabIndex();public:  __property  void
set_TabIndex(int);
[VB]      Public      Property      TabIndex      As      Integer
[JScript] public function get TabIndex() : int;public function set TabIndex(int);
```

#### *Description*

*vvvvvv)TabStop*

*wwwwww)Tag*

*xxxxxx)Text*

*yyyyyy)Top*

*zzzzzz) TopLevel*

*aaaaaaa)ToString*

#### *Description*

Gets or sets a value indicating whether to display the form as a top-level window.

A top-level form is a window that has no parent form, or whose parent form is the desktop window. Top-level windows are typically used as the main form in an application.

*bbbbbbb)TopLevelControl*

*ccccccc)TopMost*

*ddddddd)ToString*

#### *Description*

Gets or sets a value indicating whether the form should be displayed as the top-most form of your application.

A top-most form is a form that overlaps all the other forms even if it is not the active or foreground form. Top-most forms are always displayed at the highest point in the Z-order of an application. You can use this method to create a form that is always displayed in your application, such as a Find and Replace tool window.

*eeeeeee)TransparencyKey*

*ffffff) ToString*

[C#]        public        Color        TransparencyKey        {get;        set;}

[C++] public: \_\_property Color get\_TransparencyKey();public: \_\_property void set\_TransparencyKey(Color);

[VB]        Public        Property        TransparencyKey        As        Color

[JScript] public function get TransparencyKey() : Color;public function set TransparencyKey(Color);

#### *Description*

Gets or sets the color that will represent transparent areas of the form.

When the **System.Windows.Forms.Form.TransparencyKey** property is assigned a **System.Drawing.Color** , the areas of the form that have the same **System.Windows.Forms.Control.BackColor** will be displayed transparently. Any mouse actions, such as the click of the mouse, that are performed on the

transparent areas of the form will be transferred to the windows below the transparent area. For example, if the client region of a form is made transparent, clicking the mouse on that area would send the event notification of the click to any window that is below it. If the color assigned to the **System.Windows.Forms.Form.TransparencyKey** property is the same as any controls on the form, they also will be displayed transparently. For example, if you have a **System.Windows.Forms.Button** control on a form that has its **System.Windows.Forms.Form.TransparencyKey** property set to **SystemColors.Control**, the control will be displayed transparently unless the **System.Windows.Forms.Control.BackColor** property of the **System.Windows.Forms.Button** control is changed to a different color.

*ggggggg)Visible*

*hhhhhhh)VScroll*

*iiiiiii) Width*

*jjjjjjj) WindowState*

*kkkkkkk)ToString*

### *Description*

Gets or sets the form's window state.

Before a form is displayed, the **System.Windows.Forms.Form.WindowState** property is always set to **FormWindowState.Normal**, regardless of its initial setting. This is reflected in the **System.Windows.Forms.Control.Height**, **System.Windows.Forms.Control.Left**, **System.Windows.Forms.Control.Top**, and **System.Windows.Forms.Control.Width** property settings. If a form is hidden after it has been shown, these properties reflect the previous state until the form is shown again, regardless of any changes made to the **System.Windows.Forms.Form.WindowState** property.

1 **lllllll) WindowTarget**

2 **mmmmmmm)ToString**

3  
4  
5 *Description*

6 Occurs when the form is activated in code or by the user.

7 To activate a form at run time using code, call the  
8 **System.Windows.Forms.Form.Activate** method. You can use this event for  
tasks such as updating the contents of the form based on changes made to the  
form's data when the form was not activated.

9 **nnnnnnn)ToString**

10  
11  
12 *Description*

13 Occurs when the form is closed.

14 This event occurs after the form has been closed by the user or by the  
15 **System.Windows.Forms.Form.Close** method of the form. To prevent a form  
16 from closing, handle the **System.Windows.Forms.Form.Closing** event and  
set the **System.ComponentModel.CancelEventArgs.Cancel** property of the  
17 **System.ComponentModel.CancelEventArgs** passed to your event-handling  
method to **true** .

18 **oooooooo)ToString**

19  
20 [C#]        public        event        CancelEventHandler        Closing;  
21 [C++]       public:        \_\_event        CancelEventHandler\*        Closing;  
22 [VB]        Public        Event        Closing        As        CancelEventHandler

23  
24 *Description*

25 Occurs when the form is closing.

The **System.Windows.Forms.Form.Closing** event occurs as the form is being closed. When a form is closed, all resources created within the object are released and the form is disposed. If you cancel this event, the form remains opened. To cancel the closure of a form, set the **System.ComponentModel.CancelEventArgs.Cancel** property of the **System.ComponentModel.CancelEventArgs** passed to your event handler to **true**.

*ppppppp)ToString*

#### *Description*

Occurs when the form loses focus and is not the active form.

You can use this event to perform tasks such as updating another window in your application with data from the deactivated form.

*qqqqqqq)ToString*

#### *Description*

Occurs after the input language of the form has changed.

You can use this event to make changes to your form's appearance and text based on changes made to the input language of the form.

*rrrrrrr)ToString*

[C#] public event InputLanguageChangingEventHandler InputLanguageChanging;

[C++] public: \_\_event InputLanguageChangingEventHandler\*

InputLanguageChanging;

[VB] Public Event InputLanguageChanging As

InputLanguageChangingEventHandler

1  
2 *Description*

3 Occurs when the user attempts to change the input language for the form.

4 This event occurs before the change of input language is made for the form. You  
5 can cancel the language change by setting the  
6 **System.ComponentModel.CancelEventArgs.Cancel** property of the  
7 **System.Windows.Forms.InputLanguageChangingEventArgs** passed to  
your event-handling method to **false** . If the event is canceled, the input  
language is not changed. You can use this event to determine whether the  
requested input language change is appropriate for your application.

8 *sssssss)ToString*

9  
10  
11 *Description*

12 Occurs before a form is displayed for the first time.

13 You can use this event to perform tasks such as allocating resources used by the  
14 form.

15 *tttttt) ToString*

16  
17  
18 *Description*

19 Occurs when the value of the  
20 **System.Windows.Forms.Form.MaximizedBounds** property has changed.

21 For more information about handling events, see .

22 *uuuuuuu)ToString*

23 [C#] public event EventHandler MaximumSizeChanged;  
24 [C++] public: \_\_event EventHandler\* MaximumSizeChanged;  
25



1 [VB] Public Event MaximumSizeChanged As EventHandler

2  
3 *Description*

4 Occurs when the value of the **System.Windows.Forms.Form.MaximumSize**  
property has changed.

5 For more information about handling events, see .

6 *vvvvvvv)ToString*

7  
8 [C#] public event EventHandler MdiChildActivate;

9 [C++] public: \_\_event EventHandler\* MdiChildActivate;

10 [VB] Public Event MdiChildActivate As EventHandler

11  
12 *Description*

13 Occurs when a multiple document interface (MDI) child form is activated or  
14 closed within an MDI application.

15 You can use this event to perform tasks such as updating the contents of the  
MDI child form and changing the menu options available in the MDI parent form  
16 based on the status of the MDI child form that is activated.

17 *wwwwwww)ToString*

18  
19 [C#] public event EventHandler MenuComplete;

20 [C++] public: \_\_event EventHandler\* MenuComplete;

21 [VB] Public Event MenuComplete As EventHandler

22  
23 *Description*

24 Occurs when the menu of a form loses focus.

This event is raised when you click on any menu item in a menu that results in a command being performed and the menu losing focus. You can use this event to perform tasks such as updating the text of a **System.Windows.Forms.StatusBar** control or enabling and disabling buttons on a **System.Windows.Forms.ToolBar** .

*xxxxxxx)ToString*

|       |         |         |               |                 |
|-------|---------|---------|---------------|-----------------|
| [C#]  | public  | event   | EventHandler  | MenuStart;      |
| [C++] | public: | __event | EventHandler* | MenuStart;      |
| [VB]  | Public  | Event   | MenuStart     | As EventHandler |

#### *Description*

Occurs when the menu of a form receives focus.

This event is raised when any menu item in the menu is clicked by the user. You can use this event to perform tasks such as enabling and disabling controls on the form that should not be accessed by the user when the menus are being accessed.

*yyyyyyy)ToString*

|       |         |         |                    |                     |
|-------|---------|---------|--------------------|---------------------|
| [C#]  | public  | event   | EventHandler       | MinimumSizeChanged; |
| [C++] | public: | __event | EventHandler*      | MinimumSizeChanged; |
| [VB]  | Public  | Event   | MinimumSizeChanged | As EventHandler     |

#### *Description*

Occurs when the value of the **System.Windows.Forms.Form.MinimumSize** property has changed.

For more information about handling events, see .

## zzzzzzzz)Activate

|           |         |          |             |
|-----------|---------|----------|-------------|
| [C#]      | public  | void     | Activate(); |
| [C++]     | public: | void     | Activate(); |
| [VB]      | Public  | Sub      | Activate()  |
| [JScript] | public  | function | Activate(); |

### Description

Activates the form and gives it focus.

Activating a form brings it to the front if this is the active application, or it flashes the window caption if this is not the active application. The form must be visible for this method to have any effect. To determine the active form in an application, use the **System.Windows.Forms.Form.ActiveForm** property or the **System.Windows.Forms.Form.ActiveMdiChild** property if your forms are in a Multiple Document Interface (MDI) application.

## aaaaaaaa)ActivateMdiChild

|           |            |          |                        |               |
|-----------|------------|----------|------------------------|---------------|
| [C#]      | protected  | void     | ActivateMdiChild(Form  | form);        |
| [C++]     | protected: | void     | ActivateMdiChild(Form* | form);        |
| [VB]      | Protected  | Sub      | ActivateMdiChild(ByVal | form As Form) |
| [JScript] | protected  | function | ActivateMdiChild(form  | : Form);      |

### Description

This function handles the activation of a MDI child form. If a subclass overrides this function, it must call base.ActivateMdiChild This function handles the activation of a MDI child form. If a subclass overrides this function, it must call base.ActivateMdiChild Form that is getting activated

### *bbbbbbbb)AddOwnedForm*

```
1
2 [C#]      public      void      AddOwnedForm(Form      ownedForm);
3
4 [C++]      public:      void      AddOwnedForm(Form*      ownedForm);
5
6 [VB]      Public      Sub      AddOwnedForm(ByVal      ownedForm      As      Form)
7
8 [JScript]      public      function      AddOwnedForm(ownedForm      :      Form);
9
```

#### *Description*

Adds an owned form to this form.

The form assigned to the owner form remains owned until the **System.Windows.Forms.Form.RemoveOwnedForm(System.Windows.Forms.Form)** method is called. You can also make a form owned by another by setting the **System.Windows.Forms.Form.Owner** property with a reference to its owner form. The **System.Windows.Forms.Form** that this form will own.

### *ccccccc)AdjustFormScrollbars*

```
13
14
15 [C#] protected override void AdjustFormScrollbars(bool displayScrollbars);
16
17 [C++] protected: void AdjustFormScrollbars(bool displayScrollbars);
18
19 [VB] Overrides Protected Sub AdjustFormScrollbars(ByVal displayScrollbars As
20 Boolean)
21
22 [JScript] protected override function AdjustFormScrollbars(displayScrollbars :
23 Boolean);
24
```

#### *Description*

### *ddddddd)ApplyAutoScaling*

```
25 [C#]      protected      void      ApplyAutoScaling();
```

|           |            |          |                     |
|-----------|------------|----------|---------------------|
| [C++]     | protected: | void     | ApplyAutoScaling(); |
| [VB]      | Protected  | Sub      | ApplyAutoScaling()  |
| [JScript] | protected  | function | ApplyAutoScaling(); |

#### Description

This auto scales the form based on the AutoScaleBaseSize.

#### *eeeeeeee)CenterToParent*

|           |            |          |                   |
|-----------|------------|----------|-------------------|
| [C#]      | protected  | void     | CenterToParent(); |
| [C++]     | protected: | void     | CenterToParent(); |
| [VB]      | Protected  | Sub      | CenterToParent()  |
| [JScript] | protected  | function | CenterToParent(); |

#### Description

Centers the dialog to its parent.

#### *ffffffff)CenterToScreen*

|           |            |          |                   |
|-----------|------------|----------|-------------------|
| [C#]      | protected  | void     | CenterToScreen(); |
| [C++]     | protected: | void     | CenterToScreen(); |
| [VB]      | Protected  | Sub      | CenterToScreen()  |
| [JScript] | protected  | function | CenterToScreen(); |

#### Description

Centers the dialog to the screen. This will first attempt to use the owner property to determine the correct screen, then it will try the HWND owner of the form, and finally this will center the form on the same monitor as the mouse cursor.

## *gggggggg)Close*

|           |         |          |          |
|-----------|---------|----------|----------|
| [C#]      | public  | void     | Close(); |
| [C++]     | public: | void     | Close(); |
| [VB]      | Public  | Sub      | Close()  |
| [JScript] | public  | function | Close(); |

### *Description*

Closes the form.

When a form is closed, all resources created within the object are closed and the form is disposed. You can prevent the closing of a form at run time by handling the **System.Windows.Forms.Form.Closing** event and setting the **System.ComponentModel.CancelEventArgs.Cancel** property of the **System.ComponentModel.CancelEventArgs** passed as a parameter to your event-handling method. If the form you are closing is the startup form of your application, your application ends.

## *hhhhhhh)CreateControlsInstance*

|           |                                                                            |          |                    |                                               |
|-----------|----------------------------------------------------------------------------|----------|--------------------|-----------------------------------------------|
| [C#]      | protected                                                                  | override | ControlCollection  | CreateControlsInstance();                     |
| [C++]     | protected:                                                                 |          | ControlCollection* | CreateControlsInstance();                     |
| [VB]      | Overrides Protected Function CreateControlsInstance() As ControlCollection |          |                    |                                               |
| [JScript] | protected                                                                  | override | function           | CreateControlsInstance() : ControlCollection; |

### *Description*

## *iiiiiii) CreateHandle*

|       |            |          |      |                 |
|-------|------------|----------|------|-----------------|
| [C#]  | protected  | override | void | CreateHandle(); |
| [C++] | protected: |          | void | CreateHandle(); |

|           |           |           |          |                 |
|-----------|-----------|-----------|----------|-----------------|
| [VB]      | Overrides | Protected | Sub      | CreateHandle()  |
| [JScript] | protected | override  | function | CreateHandle(); |

#### *Description*

Creates the handle for the Form. If a subclass overrides this function, it must call the base implementation.

#### *jjjjjjj) DefWndProc*

|           |            |           |          |                                |
|-----------|------------|-----------|----------|--------------------------------|
| [C#]      | protected  | override  | void     | DefWndProc(ref Message m);     |
| [C++]     | protected: |           | void     | DefWndProc(Message* m);        |
| [VB]      | Overrides  | Protected | Sub      | DefWndProc(ByRef m As Message) |
| [JScript] | protected  | override  | function | DefWndProc(m : Message);       |

#### *Description*

Calls the default window proc for the form. If a subclass overrides this function, it must call the base implementation.

#### *kkkkkkkk)Dispose*

|           |            |           |          |                                     |
|-----------|------------|-----------|----------|-------------------------------------|
| [C#]      | protected  | override  | void     | Dispose(bool disposing);            |
| [C++]     | protected: |           | void     | Dispose(bool disposing);            |
| [VB]      | Overrides  | Protected | Sub      | Dispose(ByVal disposing As Boolean) |
| [JScript] | protected  | override  | function | Dispose(disposing : Boolean);       |

#### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.Form**.

Call **System.Windows.Forms.Form.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.Form**. The **System.Windows.Forms.Form.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.Form** in an unusable state. After calling **System.Windows.Forms.Form.Dispose(System.Boolean)**, you must release all references to the **System.Windows.Forms.Form** so the memory it was occupying can be reclaimed by garbage collection.

#### *lllllll) GetAutoScaleSize*

```
[C#]      public      static      SizeF      GetAutoScaleSize(Font      font);
[C++]     public:      static      SizeF      GetAutoScaleSize(Font*      font);
[VB]      Public Shared Function GetAutoScaleSize(ByVal font As Font) As SizeF
[JScrip]  public static function GetAutoScaleSize(font : Font) : SizeF;
```

#### *Description*

Gets the sized when autoscaling the form based on a specified font.

*Return Value:* A **System.Drawing.SizeF** representing the autoscaled size of the form.

You can use this method to determine the size a form would autoscale to for a specific font before applying the font to the form. If you want to determine the size a form is autoscaled to based on the form currently assigned to the form, use the **System.Windows.Forms.Form.AutoScaleBaseSize** property. A **System.Drawing.Font** representing the font to determine the autoscaled base sized of the form.

#### *mmmmmmmm)LayoutMdi*

```
[C#]      public      void      LayoutMdi(MdiLayout      value);
[C++]     public:      void      LayoutMdi(MdiLayout      value);
[VB]      Public Sub LayoutMdi(ByVal value As MdiLayout)
[JScrip]  public function LayoutMdi(value : MdiLayout);
```



## Description

Arranges the multiple document interface (MDI) child forms within the MDI parent form.

You can use this method to arrange the MDI child forms in your MDI parent form to allow for easier navigation and manipulation of MDI child forms. MDI child forms can be tiled horizontally and vertically, cascaded, or as icons within the MDI parent form. One of the **System.Windows.Forms.MdiLayout** values that defines the layout of MDI child forms.

## *nnnnnnnn)OnActivated*

[C#]       protected       virtual       void       OnActivated(EventArgs    e);

[C++]       protected:       virtual       void       OnActivated(EventArgs\*   e);

[VB]   Overridable   Protected   Sub   OnActivated(ByVal   e   As   EventArgs)

[JScript]       protected       function       OnActivated(e       :       EventArgs);

## Description

Raises the **System.Windows.Forms.Form.Activated** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## *oooooooo)OnClosed*

[C#]       protected       virtual       void       OnClosed(EventArgs    e);

[C++]       protected:       virtual       void       OnClosed(EventArgs\*   e);

[VB]   Overridable   Protected   Sub   OnClosed(ByVal   e   As   EventArgs)

[JScript]       protected       function       OnClosed(e       :       EventArgs);

*Description*

Raises the **System.Windows.Forms.Form.Closed** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

*pppppppp)OnClosing*

[C#]      protected      virtual      void      OnClosing(CancelEventArgs    e);

[C++]      protected:      virtual      void      OnClosing(CancelEventArgs\*    e);

[VB]      Overridable Protected Sub OnClosing(ByVal e As CancelEventArgs)

[JScript]      protected      function      OnClosing(e      :      CancelEventArgs);

*Description*

Raises the **System.Windows.Forms.Form.Closing** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.ComponentModel.CancelEventArgs** that contains the event data.

*qqqqqqqq)OnCreateControl*

[C#]              protected              override              void              OnCreateControl();

[C++]              protected:                              void              OnCreateControl();

[VB]              Overrides              Protected              Sub              OnCreateControl()

[JScript]              protected              override              function              OnCreateControl();

*Description*

Raises the CreateControl event.

Raising an event invokes the event-handling method through a delegate. For more information, see .

### *rrrrrrr)OnDeactivate*

[C#]      protected      virtual      void      OnDeactivate(EventArgs      e);  
[C++]      protected:      virtual      void      OnDeactivate(EventArgs\*      e);  
[VB]      Overridable      Protected      Sub      OnDeactivate(ByVal e As EventArgs)  
[JScript]      protected      function      OnDeactivate(e : EventArgs);

### *Description*

Raises the **System.Windows.Forms.Form.Deactivate** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

### *sssssss)OnFontChanged*

[C#]      protected      override      void      OnFontChanged(EventArgs      e);  
[C++]      protected:      void      OnFontChanged(EventArgs\*      e);  
[VB]      Overrides      Protected      Sub      OnFontChanged(ByVal e As EventArgs)  
[JScript]      protected      override      function      OnFontChanged(e : EventArgs);

### *Description*

### *ttttttt) OnHandleCreated*

[C#]      protected      override      void      OnHandleCreated(EventArgs      e);  
[C++]      protected:      void      OnHandleCreated(EventArgs\*      e);

1 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

2 [JScript] protected override function OnHandleCreated(e : EventArgs);

4 *Description*

5 Inheriting classes should override this method to find out when the handle has  
6 been created. Call base.OnHandleCreated first.

7 *uuuuuuuu)OnHandleDestroyed*

8 [C#] protected override void OnHandleDestroyed(EventArgs e);

9 [C++] protected: void OnHandleDestroyed(EventArgs\* e);

10 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

11 [JScript] protected override function OnHandleDestroyed(e : EventArgs);

13 *Description*

14 Inheriting classes should override this method to find out when the handle is  
15 about to be destroyed. Call base.OnHandleDestroyed last.

16 *vvvvvvvv)OnInputLanguageChanged*

18 [C#] protected virtual void

19 OnInputLanguageChanged(InputLanguageChangedEventArgs e);

20 [C++] protected: virtual void

21 OnInputLanguageChanged(InputLanguageChangedEventArgs\* e);

22 [VB] Overridable Protected Sub OnInputLanguageChanged(ByVal e As

23 InputLanguageChangedEventArgs)

24 [JScript] protected function OnInputLanguageChanged(e :

1 InputLanguageChangedEventArgs);

3 *Description*

4 Raises the **System.Windows.Forms.Form.InputLanguageChanged** event.

5 Raising an event invokes the event handler through a delegate. For more  
6 information, see . The **System.Windows.Forms.InputLanguageChangedEventArgs** that contains  
7 the event data.

8 *wwwwwwwww)OnInputLanguageChanging*

9 [C#] protected virtual void  
10 OnInputLanguageChanging(InputLanguageChangingEventArgs e);

11 [C++] protected: virtual void  
12 OnInputLanguageChanging(InputLanguageChangingEventArgs\* e);

13 [VB] Overridable Protected Sub OnInputLanguageChanging(ByVal e As  
14 InputLanguageChangingEventArgs)

15 [JScript] protected function OnInputLanguageChanging(e :  
16 InputLanguageChangingEventArgs);

18 *Description*

19 Raises the **System.Windows.Forms.Form.InputLanguageChanging** event.

20 Raising an event invokes the event handler through a delegate. For more  
21 information, see . The **System.Windows.Forms.InputLanguageChangingEventArgs** that contains  
22 the event data.

### xxxxxxx)OnLoad

```
[C#]      protected      virtual      void      OnLoad(EventArgs      e);
[C++]      protected:      virtual      void      OnLoad(EventArgs*      e);
[VB]      Overridable      Protected      Sub      OnLoad(ByVal e As EventArgs)
[JScript]      protected      function      OnLoad(e      :      EventArgs);
```

#### Description

Raises the **System.Windows.Forms.Form.Load** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### yyyyyyyy)OnMaximizedBoundsChanged

```
[C#]      protected      virtual      void      OnMaximizedBoundsChanged(EventArgs e);
[C++]      protected:      virtual      void      OnMaximizedBoundsChanged(EventArgs* e);
[VB]      Overridable      Protected      Sub      OnMaximizedBoundsChanged(ByVal e As
EventArgs)
[JScript]      protected      function      OnMaximizedBoundsChanged(e : EventArgs);
```

#### Description

Raises the **System.Windows.Forms.Form.MaximizedBoundsChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

## *zzzzzzzz)OnMaximumSizeChanged*

```
1
2
3 [C#] protected virtual void OnMaximumSizeChanged(EventArgs e);
4 [C++] protected: virtual void OnMaximumSizeChanged(EventArgs* e);
5 [VB] Overridable Protected Sub OnMaximumSizeChanged(ByVal e As
6 EventArgs)
7 [JScript] protected function OnMaximumSizeChanged(e : EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Form.MaximumSizeChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

## *aaaaaaaaa)OnMdiChildActivate*

```
13
14 [C#] protected virtual void OnMdiChildActivate(EventArgs e);
15 [C++] protected: virtual void OnMdiChildActivate(EventArgs* e);
16 [VB] Overridable Protected Sub OnMdiChildActivate(ByVal e As EventArgs)
17 [JScript] protected function OnMdiChildActivate(e : EventArgs);
```

### *Description*

Raises the **System.Windows.Forms.Form.MdiChildActivate** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

## *bbbbbbbbbb)OnMenuComplete*

```
23
24
25 [C#] protected virtual void OnMenuComplete(EventArgs e);
```

```

1 [C++] protected: virtual void OnMenuComplete(EventArgs* e);
2 [VB] Overridable Protected Sub OnMenuComplete(ByVal e As EventArgs)
3 [JScript] protected function OnMenuComplete(e : EventArgs);

```

#### *Description*

Raises the **System.Windows.Forms.Form.MenuComplete** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

#### *cccccccc)OnMenuStart*

```

10 [C#] protected virtual void OnMenuStart(EventArgs e);
11 [C++] protected: virtual void OnMenuStart(EventArgs* e);
12 [VB] Overridable Protected Sub OnMenuStart(ByVal e As EventArgs)
13 [JScript] protected function OnMenuStart(e : EventArgs);

```

#### *Description*

Raises the **System.Windows.Forms.Form.MenuStart** event.

Raising an event invokes the event handler through a delegate. For more information, see . The **System.EventArgs** that contains the event data.

#### *dddddddd)OnMinimumSizeChanged*

```

21 [C#] protected virtual void OnMinimumSizeChanged(EventArgs e);
22 [C++] protected: virtual void OnMinimumSizeChanged(EventArgs* e);
23 [VB] Overridable Protected Sub OnMinimumSizeChanged(ByVal e As
24 EventArgs)
25 [JScript] protected function OnMinimumSizeChanged(e : EventArgs);

```



## Description

Raises the **System.Windows.Forms.Form.MinimumSizeChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## eeeeeeee)OnPaint

[C#]      protected      override      void      OnPaint(PaintEventArgs      e);

[C++]      protected:      void      OnPaint(PaintEventArgs\*      e);

[VB]      Overrides      Protected      Sub      OnPaint(ByVal      e      As      PaintEventArgs)

[JScript]      protected      override      function      OnPaint(e      :      PaintEventArgs);

## Description

Raises the Paint event.

## ffffffff)OnResize

[C#]      protected      override      void      OnResize(EventArgs      e);

[C++]      protected:      void      OnResize(EventArgs\*      e);

[VB]      Overrides      Protected      Sub      OnResize(ByVal      e      As      EventArgs)

[JScript]      protected      override      function      OnResize(e      :      EventArgs);

## Description

Raises the Resize event.

### *ggggggggg)OnStyleChanged*

```
1
2
3 [C#]    protected    override    void    OnStyleChanged(EventArgs    e);
4
5 [C++]    protected:    void    OnStyleChanged(EventArgs*    e);
6
7 [VB] Overrides Protected Sub OnStyleChanged(ByVal e As EventArgs)
8
9 [JScript] protected override function OnStyleChanged(e : EventArgs);
```

#### *Description*

Raises the StyleChanged event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### *hhhhhhhhh)OnTextChanged*

```
12
13 [C#]    protected    override    void    OnTextChanged(EventArgs    e);
14
15 [C++]    protected:    void    OnTextChanged(EventArgs*    e);
16
17 [VB] Overrides Protected Sub OnTextChanged(ByVal e As EventArgs)
18
19 [JScript] protected override function OnTextChanged(e : EventArgs);
```

#### *Description*

### *iiiiiii) OnVisibleChanged*

```
21
22 [C#]    protected    override    void    OnVisibleChanged(EventArgs    e);
23
24 [C++]    protected:    void    OnVisibleChanged(EventArgs*    e);
25
26 [VB] Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)
```

1 [JScript] protected override function OnVisibleChanged(e : EventArgs);

3 *Description*

4 Raises the **System.Windows.Forms.Control.VisibleChanged** event.

5 Raising an event invokes the event handler through a delegate. For more  
6 information, see . The **System.EventArgs** that contains the event data.

7 *jjjjjjjj) ProcessCmdKey*

8 [C#] protected override bool ProcessCmdKey(ref Message msg, Keys keyData);

9 [C++] protected: bool ProcessCmdKey(Message\* msg, Keys keyData);

10 [VB] Overrides Protected Function ProcessCmdKey(ByRef msg As Message,

11 ByVal keyData As Keys) As Boolean

12 [JScript] protected override function ProcessCmdKey(msg : Message, keyData :  
13 Keys) : Boolean;

15 *Description*

16 Processes a command key. Overrides Control.processCmdKey() to provide  
17 additional handling of main menu command keys and Mdi accelerators.

18 *Return Value:* true to consume the key, false to allow further processing. original  
19 window message. key code and modifier flags.

20 *kkkkkkkkk) ProcessDialogKey*

21 [C#] protected override bool ProcessDialogKey(Keys keyData);

22 [C++] protected: bool ProcessDialogKey(Keys keyData);

23 [VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)

24 As Boolean

1 [JScript] protected override function ProcessDialogKey(keyData : Keys) :  
2 Boolean;

3  
4 *Description*

5 Processes a dialog key. Overrides Control.processDialogKey(). This method  
6 implements handling of the RETURN, and ESCAPE keys in dialogs. The method  
7 performs no processing on keys that include the ALT or CONTROL modifiers.  
8 *Return Value:* true to consume the key, false to allow further processing. key  
9 code and modifier flags.

8 *|||||||) ProcessKeyPreview*

10 [C#] protected override bool ProcessKeyPreview(ref Message m);  
11 [C++] protected: bool ProcessKeyPreview(Message\* m);  
12 [VB] Overrides Protected Function ProcessKeyPreview(ByRef m As Message) As  
13 Boolean  
14 [JScript] protected override function ProcessKeyPreview(m : Message) : Boolean;

16 *Description*

18 *mmmmmmmmmm)ProcessTabKey*

20 [C#] protected override bool ProcessTabKey(bool forward);  
21 [C++] protected: bool ProcessTabKey(bool forward);  
22 [VB] Overrides Protected Function ProcessTabKey(ByVal forward As Boolean)  
23 As Boolean  
24 [JScript] protected override function ProcessTabKey(forward : Boolean) :  
25

Boolean;

*Description*

*nnnnnnnnnn)RemoveOwnedForm*

[C#]      public      void      RemoveOwnedForm(Form      ownedForm);

[C++]      public:      void      RemoveOwnedForm(Form\*      ownedForm);

[VB]      Public      Sub      RemoveOwnedForm(ByVal      ownedForm      As      Form)

[JScript]      public      function      RemoveOwnedForm(ownedForm      :      Form);

*Description*

Removes an owned form from this form.

he form assigned to the owner form remains owned until the **System.Windows.Forms.Form.RemoveOwnedForm(System.Windows.Forms.Form)** method is called. In addition to removing the owned form from the list of owned form, this method also sets the owner form to **null** . A **System.Windows.Forms.Form** representing the form to remove from the list of owned forms for this form.

*ooooooooo)ScaleCore*

[C#]      protected      override      void      ScaleCore(float      x,      float      y);

[C++]      protected:      void      ScaleCore(float      x,      float      y);

[VB]      Overrides Protected Sub ScaleCore(ByVal x As Single, ByVal y As Single)

[JScript]      protected      override      function      ScaleCore(x      :      float,      y      :      float);

*Description*

Base function that performs scaling of the form. Percentage to scale the form horizontally Percentage to scale the form vertically

*pppppppppp)Select*

[C#] protected override void Select(bool directed, bool forward);  
[C++] protected: void Select(bool directed, bool forward);  
[VB] Overrides Protected Sub Select(ByVal directed As Boolean, ByVal forward As Boolean)  
[JScript] protected override function Select(directed : Boolean, forward : Boolean);

*Description*

Selects this form, and optionally selects the next/previous control. If set to true that the active control is changed If directed is true, then this controls the direction in which focus is moved. If this is true, then the next control is selected, otherwise the previous control is selected.

*qqqqqqqqq)SetBoundsCore*

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);  
[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);  
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)  
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

1  
2 *Description*

3 *rrrrrrrrr)SetClientSizeCore*

4  
5 [C#] protected override void SetClientSizeCore(int x, int y);

6 [C++] protected: void SetClientSizeCore(int x, int y);

7 [VB] Overrides Protected Sub SetClientSizeCore(ByVal x As Integer, ByVal y As  
8 Integer)

9 [JScript] protected override function SetClientSizeCore(x : int, y : int);

10  
11 *Description*

12 Sets the clientSize of the form. This will adjust the bounds of the form to make  
13 the clientSize the requested size. Requested width of the client region Requested  
height of the client region

14 *sssssssss)SetDesktopBounds*

15  
16 [C#] public void SetDesktopBounds(int x, int y, int width, int height);

17 [C++] public: void SetDesktopBounds(int x, int y, int width, int height);

18 [VB] Public Sub SetDesktopBounds(ByVal x As Integer, ByVal y As Integer,  
19 ByVal width As Integer, ByVal height As Integer)

20 [JScript] public function SetDesktopBounds(x : int, y : int, width : int, height :  
21 int);

22  
23 *Description*

24 Sets the bounds of the form in desktop coordinates.

Desktop coordinates are based on the working area of the screen, which excludes the taskbar. You can use this method to set the position and size of your form on the desktop. Since desktop coordinates are based on the working area of the form, you can use this method to ensure that your form is completely visible on the desktop. The x-coordinate of the form's location. The y-coordinate of the form's location. The width of the form. The height of the form.

#### *tttttttt) SetDesktopLocation*

```
[C#]      public      void      SetDesktopLocation(int      x,      int      y);  
[C++]     public:     void      SetDesktopLocation(int      x,      int      y);  
[VB] Public Sub SetDesktopLocation(ByVal x As Integer, ByVal y As Integer)  
[JScript] public  function  SetDesktopLocation(x : int, y : int);
```

#### *Description*

Sets the location of the form in desktop coordinates.

Desktop coordinates are based on the working area of the screen, which excludes the taskbar. You can use this method to position your form on the desktop. Since desktop coordinates are based on the working area of the form, you can use this method to ensure that your form is completely visible on the desktop. The x-coordinate of the form's location. The y-coordinate of the form's location.

#### *uuuuuuuuuu) SetVisibleCore*

```
[C#]      protected  override  void      SetVisibleCore(bool      value);  
[C++]     protected:          void      SetVisibleCore(bool      value);  
[VB] Overrides Protected Sub SetVisibleCore(ByVal value As Boolean)  
[JScript] protected override function SetVisibleCore(value : Boolean);
```



## vvvvvvvvv)ShowDialog

```
1
2 [C#]          public          DialogResult          ShowDialog();
3 [C++]          public:          DialogResult          ShowDialog();
4 [VB]          Public          Function          ShowDialog()          As          DialogResult
5 [JScript] public function ShowDialog() : DialogResult; Shows the form as a modal
6 dialog                                              box.
```

### Description

Shows the form as a modal dialog box with no owner window.

**Return Value:** One of the **System.Windows.Forms.DialogResult** values.

You can use this method to display a modal dialog box in your application. When this method is called, the code following it is not executed until after the dialog box is closed. The dialog box can be assigned one of the values of the **System.Windows.Forms.DialogResult** enumeration by assigning it to the **System.Windows.Forms.Button.DialogResult** property of a **System.Windows.Forms.Button** on the form or by setting the **System.Windows.Forms.Form.DialogResult** property of the form in code. This value is then returned by this method. You can use this return value to determine how to process the actions that occurred in the dialog box. For example, if the dialog box was closed and returned the **DialogResult.Cancel** value through this method, you could prevent code following the call to **System.Windows.Forms.Form.ShowDialog** from executing.

## wwwwwwwww)ShowDialog

```
19
20 [C#]          public          DialogResult          ShowDialog(IWin32Window          owner);
21 [C++]          public:          DialogResult          ShowDialog(IWin32Window*          owner);
22 [VB]          Public          Function          ShowDialog(ByVal          owner          As          IWin32Window)          As
23 DialogResult
24 [JScript] public function ShowDialog(owner : IWin32Window) : DialogResult;
```

## Description

Shows the form as a modal dialog with the specified owner.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can use this method to display a modal dialog box in your application. When this method is called, the code following it is not executed until after the dialog box is closed. The dialog box can be assigned one of the values of **System.Windows.Forms.DialogResult** by assigning it to the **System.Windows.Forms.Button.DialogResult** property of a **System.Windows.Forms.Button** on the form or by setting the **System.Windows.Forms.Form.DialogResult** property of the form in code. This value is then returned by this method. You can use this return value to determine how to process the actions that occurred in the dialog box. For example, if the dialog box was closed and returned the **DialogResult.Cancel** value through this method, you could prevent code following the call to **System.Windows.Forms.Form.ShowDialog** from executing. Any object that implements **System.Windows.Forms.IWin32Window** that represents the top-level window that will own the modal dialog.

*xxxxxxxxxx)ToString*

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

## Description

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

*yyyyyyyyyy)UpdateDefaultButton*

|      |           |          |      |                        |
|------|-----------|----------|------|------------------------|
| [C#] | protected | override | void | UpdateDefaultButton(); |
|------|-----------|----------|------|------------------------|

|   |           |            |                   |                        |
|---|-----------|------------|-------------------|------------------------|
| 1 | [C++]     | protected: | void              | UpdateDefaultButton(); |
| 2 | [VB]      | Overrides  | Protected Sub     | UpdateDefaultButton()  |
| 3 | [JScript] | protected  | override function | UpdateDefaultButton(); |

4

5 *Description*

6 Updates the default button based on current selection, and the acceptButton property.

7 *zzzzzzzzzz)WndProc*

|    |           |            |               |                    |                  |         |           |
|----|-----------|------------|---------------|--------------------|------------------|---------|-----------|
| 9  | [C#]      | protected  | override      | void               | WndProc(ref      | Message | m);       |
| 10 | [C++]     | protected: |               | void               | WndProc(Message* | m);     |           |
| 11 | [VB]      | Overrides  | Protected Sub | WndProc(ByRef m As | Message)         |         |           |
| 12 | [JScript] | protected  | override      | function           | WndProc(m        | :       | Message); |

14

15 *Description*

16 Base wndProc encapsulation. Message sent

17 DataFormats.Format class (System.Windows.Forms)

18 *a) WndProc*

20

21 *Description*

22 Represents a clipboard format type.

23 A format type consists of a text-based format name and an ID number. The format name/ID number pair can define a system

24 **System.Windows.Forms.Clipboard** or other format.

b) *DataFormats.Format*

*Example Syntax:*

c) *WndProc*

```
[C#]      public      DataFormats.Format(string      name,      int      id);  
[C++]      public:      Format(String*      name,      int      id);  
[VB] Public Sub New(ByVal name As String, ByVal id As Integer)  
[JScript] public function DataFormats.Format(name : String, id : int);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.DataFormats.Format** class with a Boolean that indicates whether a **Win32** handle is expected. The name of this format. The ID number for this format.

d) *Id*

e) *WndProc*

```
[C#]      public      int      Id      {get;}  
[C++]      public:      __property      int      get_Id();  
[VB] Public ReadOnly Property Id As Integer  
[JScript] public function get Id() : int;
```

*Description*

Gets the ID number for this format.

*f) Name*

*g) WndProc*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | string     | Name     | {get;}           |
| [C++]     | public: | __property | String*  | get_Name();      |
| [VB]      | Public  | ReadOnly   | Property | Name As String   |
| [JScript] | public  | function   | get      | Name() : String; |

*Description*

Gets the name of this format.

FormBorderStyle enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the border styles for a form.

This enumeration is used by the **System.Windows.Forms.Form** class.

*b) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | Fixed3D;           |
| [C++]     | public: | const | FormBorderStyle | Fixed3D;           |
| [VB]      | Public  | Const | Fixed3D         | As FormBorderStyle |
| [JScript] | public  | var   | Fixed3D         | : FormBorderStyle; |

*Description*

A fixed, three-dimensional border.

*c) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | FixedDialog;       |
| [C++]     | public: | const | FormBorderStyle | FixedDialog;       |
| [VB]      | Public  | Const | FixedDialog     | As FormBorderStyle |
| [JScript] | public  | var   | FixedDialog     | : FormBorderStyle; |

*Description*

A thick, fixed dialog-style border.

*d) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | FixedSingle;       |
| [C++]     | public: | const | FormBorderStyle | FixedSingle;       |
| [VB]      | Public  | Const | FixedSingle     | As FormBorderStyle |
| [JScript] | public  | var   | FixedSingle     | : FormBorderStyle; |

*Description*

A fixed, single line border.

*e) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | FixedToolWindow;   |
| [C++]     | public: | const | FormBorderStyle | FixedToolWindow;   |
| [VB]      | Public  | Const | FixedToolWindow | As FormBorderStyle |
| [JScript] | public  | var   | FixedToolWindow | : FormBorderStyle; |

*Description*

A tool window border that is not resizable.

*f) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | None;              |
| [C++]     | public: | const | FormBorderStyle | None;              |
| [VB]      | Public  | Const | None            | As FormBorderStyle |
| [JScript] | public  | var   | None            | : FormBorderStyle; |

*Description*

No border.

*g) ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormBorderStyle | Sizable;           |
| [C++]     | public: | const | FormBorderStyle | Sizable;           |
| [VB]      | Public  | Const | Sizable         | As FormBorderStyle |
| [JScript] | public  | var   | Sizable         | : FormBorderStyle; |

*Description*

A resizable border.

*h) ToString*

|       |         |       |                 |                    |
|-------|---------|-------|-----------------|--------------------|
| [C#]  | public  | const | FormBorderStyle | SizableToolWindow; |
| [C++] | public: | const | FormBorderStyle | SizableToolWindow; |

1 [VB] Public Const SizableToolWindow As FormBorderStyle

2 [JScript] public var SizableToolWindow : FormBorderStyle;

4 *Description*

5 A resizable tool window border.

6 FormStartPosition enumeration (System.Windows.Forms)

7 *a) ToString*

10 *Description*

11 Specifies the initial position of a form.

12 This enumeration is used by the **System.Windows.Forms.Form** class.

13 *b) ToString*

15 [C#] public const FormStartPosition CenterParent;

16 [C++] public: const FormStartPosition CenterParent;

17 [VB] Public Const CenterParent As FormStartPosition

18 [JScript] public var CenterParent : FormStartPosition;

20 *Description*

21 The form is centered within the bounds of its parent form.

22 *c) ToString*

24 [C#] public const FormStartPosition CenterScreen;

25 [C++] public: const FormStartPosition CenterScreen;



```

1  [VB]      Public      Const      CenterScreen      As      FormStartPosition
2  [JScript]      public      var      CenterScreen      :      FormStartPosition;

```

#### *Description*

The form is centered on the current display, and has the dimensions specified in the form's size.

#### *d) ToString*

```

8  [C#]      public      const      FormStartPosition      Manual;
9  [C++]      public:      const      FormStartPosition      Manual;
10 [VB]      Public      Const      Manual      As      FormStartPosition
11 [JScript]      public      var      Manual      :      FormStartPosition;

```

#### *Description*

The location and size of the form will determine its starting position.

#### *e) ToString*

```

17 [C#]      public      const      FormStartPosition      WindowsDefaultBounds;
18 [C++]      public:      const      FormStartPosition      WindowsDefaultBounds;
19 [VB]      Public      Const      WindowsDefaultBounds      As      FormStartPosition
20 [JScript]      public      var      WindowsDefaultBounds      :      FormStartPosition;

```

#### *Description*

The form is positioned at the Windows default location and has the bounds determined by Windows default.

*f) ToString*

```
[C#]    public    const    FormStartPosition    WindowsDefaultLocation;  
[C++]   public:   const    FormStartPosition    WindowsDefaultLocation;  
[VB]    Public    Const    WindowsDefaultLocation    As    FormStartPosition  
[JScript] public    var    WindowsDefaultLocation    :    FormStartPosition;
```

*Description*

The form is positioned at the Windows default location and has the dimensions specified in the form's size.

FormWindowState enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies how a form window is displayed.

This enumeration is used by the **System.Windows.Forms.Form** class.

*b) ToString*

```
[C#]    public    const    FormWindowState    Maximized;  
[C++]   public:   const    FormWindowState    Maximized;  
[VB]    Public    Const    Maximized    As    FormWindowState  
[JScript] public    var    Maximized    :    FormWindowState;
```

*Description*

A maximized window.

c) *ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormWindowState | Minimized;         |
| [C++]     | public: | const | FormWindowState | Minimized;         |
| [VB]      | Public  | Const | Minimized       | As FormWindowState |
| [JScript] | public  | var   | Minimized       | : FormWindowState; |

*Description*

A minimized window.

d) *ToString*

|           |         |       |                 |                    |
|-----------|---------|-------|-----------------|--------------------|
| [C#]      | public  | const | FormWindowState | Normal;            |
| [C++]     | public: | const | FormWindowState | Normal;            |
| [VB]      | Public  | Const | Normal          | As FormWindowState |
| [JScript] | public  | var   | Normal          | : FormWindowState; |

*Description*

A default sized window.

FrameStyle enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies the frame style of the selected control.

This enumeration is used by members such as

**System.Windows.Forms.ControlPaint.DrawReversibleFrame(System.Dr**

awing.Rectangle, System.Drawing.Color, System.Windows.Forms.Frame  
Style) .

*b) ToString*

|           |         |       |            |               |
|-----------|---------|-------|------------|---------------|
| [C#]      | public  | const | FrameStyle | Dashed;       |
| [C++]     | public: | const | FrameStyle | Dashed;       |
| [VB]      | Public  | Const | Dashed     | As FrameStyle |
| [JScript] | public  | var   | Dashed     | : FrameStyle; |

*Description*

A thin, dashed border.

*c) ToString*

|           |         |       |            |               |
|-----------|---------|-------|------------|---------------|
| [C#]      | public  | const | FrameStyle | Thick;        |
| [C++]     | public: | const | FrameStyle | Thick;        |
| [VB]      | Public  | Const | Thick      | As FrameStyle |
| [JScript] | public  | var   | Thick      | : FrameStyle; |

*Description*

A thick, solid border.

GiveFeedbackEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **System.Windows.Forms.Control.GiveFeedback** event.

The **System.Windows.Forms.Control.GiveFeedback** event occurs during a drag operation. It allows the source of a drag event to modify the appearance of the mouse pointer in order to give the user visual feedback during a drag-and-drop operation. A **System.Windows.Forms.GiveFeedbackEventArgs** object specifies the type of drag-and-drop operation and whether default cursors are used.

b) *GiveFeedbackEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#] public GiveFeedbackEventArgs(DragDropEffects effect, bool
useDefaultCursors);
```

```
[C++] public: GiveFeedbackEventArgs(DragDropEffects effect, bool
useDefaultCursors);
```

```
[VB] Public Sub New(ByVal effect As DragDropEffects, ByVal
useDefaultCursors As Boolean)
```

```
[JScript] public function GiveFeedbackEventArgs(effect : DragDropEffects,
useDefaultCursors : Boolean);
```

### *Description*

Initializes a new instance of the **System.Windows.Forms.GiveFeedbackEventArgs** class. The type of drag-and-drop operation. Possible values are obtained by applying the bitwise OR (|) operation to the constants defined in the **System.Windows.Forms.DragDropEffects**. **true** if default pointers are used; otherwise, **false**.

1        **d)     *Effect***

2        **e)     *ToString***

3  
4    [C#]        public        DragDropEffects        Effect        {get;}

5    [C++]        public:        \_\_property        DragDropEffects        get\_Effect();

6    [VB]        Public        ReadOnly        Property        Effect        As        DragDropEffects

7    [JScript]    public        function        get        Effect()        :        DragDropEffects;

8  
9        *Description*

10       Gets the type of drag-and-drop operation.

11       **f)     *UseDefaultCursors***

12       **g)     *ToString***

13  
14    [C#]        public        bool        UseDefaultCursors        {get;        set;}

15    [C++]        public:        \_\_property bool get\_UseDefaultCursors();public:        \_\_property void  
16    set\_UseDefaultCursors(bool);

17    [VB]        Public        Property        UseDefaultCursors        As        Boolean

18    [JScript]    public function get UseDefaultCursors() : Boolean;public function set  
19    UseDefaultCursors(Boolean);

20  
21       *Description*

22       Gets or sets whether the default pointer is used.

23       The default pointer is usually an arrow. A pointer is sometimes called a cursor.

GiveFeedbackEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that handles the **System.Windows.Forms.Control.GiveFeedback** event of a **System.Windows.Forms.Control** . The source of the event. A **System.Windows.Forms.GiveFeedbackEventArgs** that contains the event data.

When you create a **System.Windows.Forms.GiveFeedbackEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

GridColumnStylesCollection class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a collection of **System.Windows.Forms.DataGridColumnStyle** objects in the **System.Windows.Forms.DataGrid** control.

On the **System.Windows.Forms.DataGridTableStyle** , you access the **System.Windows.Forms.GridColumnStylesCollection** through the **System.Windows.Forms.DataGridTableStyle.GridColumnStyles** property.

1           **b)     Count**

2           **c)     IsReadOnly**

3           **d)     IsSynchronized**

4           **e)     Item**

5           **f)     ToString**

6           **System.Windows.Forms.DataGridColumnStyleSystem.Windows.Form**

7           **s.GridColumnStylesCollection**

8  
9           *Description*

10          Gets the **System.Windows.Forms.DataGridColumnStyle** at a specified  
index.

11          Use the  
12          **System.Windows.Forms.GridColumnStylesCollection.IndexOf(System.**  
13          **Windows.Forms.DataGridColumnStyle)** method to determine the index of  
any element in the collection. The zero-based index of the  
14          **System.Windows.Forms.DataGridColumnStyle** to return.

15           **g)     Item**

16           **h)     ToString**

17  
18          [C#]    public    DataGridColumnStyle    this[string    columnName]    {get;}

19          [C++]   public: \_\_property DataGridColumnStyle\* get\_Item(String\* columnName);

20          [VB]   Public Default ReadOnly Property Item(ByVal columnName As String) As  
21          DataGridColumnStyle

22          [JScript]   returnValue = GridColumnStylesCollectionObject.Item(columnName);

23  
24           *Description*

25



Gets the **System.Windows.Forms.DataGridColumnStyle** with the specified name.

The column header of a **System.Windows.Forms.DataGridColumnStyle** can be set explicitly by setting the **System.Windows.Forms.DataGridColumnStyle.HeaderText** property. By default, the **System.Windows.Forms.DataGridColumnStyle.HeaderText** is set using uses **System.Windows.Forms.DataGridColumnStyle.MappingName** property value. The **System.Windows.Forms.DataGridColumnStyle.MappingName** of the **System.Windows.Forms.DataGridColumnStyle** retrieve.

i) *Item*

j) *ToString*

[C#] public DataGridColumnStyle this[PropertyDescriptor propDesc] {get;}

[C++] public: \_\_property DataGridColumnStyle\* get\_Item(PropertyDescriptor\* propDesc);

[VB] Public Default ReadOnly Property Item(ByVal propDesc As  
PropertyDescriptor) As DataGridColumnStyle

[JScript] returnValue = GridColumnStylesCollectionObject.Item(propDesc);

### *Description*

Gets the **System.Windows.Forms.DataGridColumnStyle** associated with the specified **System.ComponentModel.PropertyDescriptor** .

Each **System.Windows.Forms.DataGridColumnStyle** object is created using a **System.ComponentModel.PropertyDescriptor** . The **System.ComponentModel.PropertyDescriptor** can be returned using the **System.Windows.Forms.DataGridColumnStyle.PropertyDescriptor** property. The **System.ComponentModel.PropertyDescriptor** associated with the **System.Windows.Forms.DataGridColumnStyle**.

### l) *ToString*

|           |            |            |           |            |                   |
|-----------|------------|------------|-----------|------------|-------------------|
| [C#]      | protected  | override   | ArrayList | List       | {get;}            |
| [C++]     | protected: | __property | virtual   | ArrayList* | get_List();       |
| [VB]      | Overrides  | Protected  | ReadOnly  | Property   | List As ArrayList |
| [JScript] | protected  | function   | get       | List()     | : ArrayList;      |

### Description

Gets the list of items in the collection.

*m) SyncRoot*

***n) ToString***

*Description*

Occurs when a change is made to the **System.Windows.Forms.GridColumnStylesCollection**.

For more information about handling events, see .

***o) Add***

|           |                                                                      |          |                                   |                          |          |
|-----------|----------------------------------------------------------------------|----------|-----------------------------------|--------------------------|----------|
| [C#]      | public                                                               | virtual  | int                               | Add(DataGridColumnStyle  | column); |
| [C++]     | public:                                                              | virtual  | int                               | Add(DataGridColumnStyle* | column); |
| [VB]      | Overridable Public Function Add(ByVal column As DataGridColumnStyle) |          |                                   |                          |          |
|           | As                                                                   |          |                                   |                          | Integer  |
| [JScript] | public                                                               | function | Add(column : DataGridColumnStyle) | :                        | int;     |

## Description

Adds a column style to the collection.

*Return Value:* The index of the new

**System.Windows.Forms.DataGridColumnStyle** object. The **System.Windows.Forms.DataGridColumnStyle** to add.

### p) AddRange

[C#] public void AddRange(DataGridColumnStyle[] columns);

[C++] public: void AddRange(DataGridColumnStyle\* columns[]);

[VB] Public Sub AddRange(ByVal columns() As DataGridColumnStyle)

[JScript] public function AddRange(columns : DataGridColumnStyle[]);

## Description

Adds an array of column style objects to the collection. An array of **System.Windows.Forms.DataGridColumnStyle** objects to add to the collection.

### q) Clear

[C#] public void Clear();

[C++] public: void Clear();

[VB] Public Sub Clear()

[JScript] public function Clear();

## Description

Clears the collection of **System.Windows.Forms.DataGridColumnStyle** objects.

*r) Contains*

```
[C#]      public      bool      Contains(DataGridColumnStyle      column);  
[C++]     public:     bool      Contains(DataGridColumnStyle*      column);  
[VB] Public Function Contains(ByVal column As DataGridColumnStyle) As  
Boolean  
[JScript] public function Contains(column : DataGridColumnStyle) : Boolean;
```

*Description*

Gets a value indicating whether the **System.Windows.Forms.GridColumnStylesCollection** contains the specified **System.Windows.Forms.DataGridColumnStyle**.

*Return Value:* **true** if the collection contains the **System.Windows.Forms.DataGridColumnStyle**; otherwise, **false**.

Use the **System.Windows.Forms.GridColumnStylesCollection.Contains(System.ComponentModel.PropertyDescriptor)** method to determine if a particular **System.Windows.Forms.DataGridColumnStyle** object exists before invoking the **System.Windows.Forms.GridColumnStylesCollection.Remove(System.Windows.Forms.DataGridColumnStyle)** method to remove the item. If you need to know the index of a particular **System.Windows.Forms.DataGridColumnStyle**, use the **System.Windows.Forms.GridColumnStylesCollection.IndexOf(System.Windows.Forms.DataGridColumnStyle)** method. The desired **System.Windows.Forms.DataGridColumnStyle**.

*s) Contains*

```
[C#]      public      bool      Contains(PropertyDescriptor      propDesc);  
[C++]     public:     bool      Contains(PropertyDescriptor*      propDesc);  
[VB] Public Function Contains(ByVal propDesc As PropertyDescriptor) As  
Boolean
```

[JScript] public function Contains(propDesc : PropertyDescriptor) : Boolean; Gets a value indicating whether the **System.Windows.Forms.GridColumnStylesCollection** contains a specific **System.Windows.Forms.DataGridColumnStyle**.

#### *Description*

Gets a value indicating whether the **System.Windows.Forms.GridColumnStylesCollection** contains a **System.Windows.Forms.DataGridColumnStyle** associated with the specified **System.ComponentModel.PropertyDescriptor**.

*Return Value:* **true** if the collection contains the **System.Windows.Forms.DataGridColumnStyle**; otherwise, **false**.

To get a **System.ComponentModel.PropertyDescriptorCollection**, use the **System.Windows.Forms.BindingManagerBase.GetItemProperties** method of the **System.Windows.Forms.BindingManagerBase** class. Pass the **System.Windows.Forms.DataGridColumnStyle.MappingName** of the **System.Windows.Forms.DataGridColumnStyle** to the **System.ComponentModel.PropertyDescriptorCollection.Item(System.Int32)** property of the **System.ComponentModel.PropertyDescriptorCollection** to return the **System.ComponentModel.PropertyDescriptor** for a specific column. The **System.ComponentModel.PropertyDescriptor** associated with the desired **System.Windows.Forms.DataGridColumnStyle**.

#### *t) Contains*

|           |                 |                |                         |            |
|-----------|-----------------|----------------|-------------------------|------------|
| [C#]      | public          | bool           | Contains(string         | name);     |
| [C++]     | public:         | bool           | Contains(String*        | name);     |
| [VB]      | Public Function | Contains(ByVal | name As String)         | As Boolean |
| [JScript] | public          | function       | Contains(name : String) | : Boolean; |

#### *Description*

Gets a value indicating whether the

**System.Windows.Forms.GridColumnStylesCollection** contains the **System.Windows.Forms.DataGridColumnStyle** with the specified name.

*Return Value:* **true** if the collection contains the

**System.Windows.Forms.DataGridColumnStyle** ; otherwise, **false** .

The caption of a **System.Windows.Forms.DataGridColumnStyle** is set with the **System.Windows.Forms.DataGridColumnStyle.HeaderText** property. The **System.Windows.Forms.DataGridColumnStyle.MappingName** of the desired **System.Windows.Forms.DataGridColumnStyle**.

*u) IndexOf*

[C#]        public        int        IndexOf(DataGridColumnStyle        element);

[C++]        public:        int        IndexOf(DataGridColumnStyle\*        element);

[VB] Public Function IndexOf(ByVal element As DataGridColumnStyle) As Integer

[JScript] public function IndexOf(element : DataGridColumnStyle) : int;

*Description*

Gets the index of a specified

**System.Windows.Forms.DataGridColumnStyle** .

*Return Value:* The zero-based index of the

**System.Windows.Forms.DataGridColumnStyle** within the

**System.Windows.Forms.GridColumnStylesCollection** or -1 if no

corresponding **System.Windows.Forms.DataGridColumnStyle** exists.

Use the

**System.Windows.Forms.GridColumnStylesCollection.Contains(System.ComponentModel.PropertyDescriptor)** method to determine if a specific

**System.Windows.Forms.DataGridColumnStyle** exists. If so, and you need the index of the element within the collection, use the

**System.Windows.Forms.GridColumnStylesCollection.IndexOf(System.Windows.Forms.DataGridColumnStyle)** method. The

**System.Windows.Forms.DataGridColumnStyle** to find.

v) *OnCollectionChanged*

```
[C#] protected void OnCollectionChanged(CollectionChangeEventArgs ccevent);  
[C++] protected: void OnCollectionChanged(CollectionChangeEventArgs*  
ccevent);  
[VB] Protected Sub OnCollectionChanged(ByVal ccevent As  
CollectionChangeEventArgs)  
[JScript] protected function OnCollectionChanged(ccevent :  
CollectionChangeEventArgs);
```

*Description*

Raises the **System.Windows.Forms.GridColumnStylesCollection.CollectionChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.ComponentModel.CollectionChangeEventArgs** that contains the event data event.

w) *Remove*

```
[C#] public void Remove(DataGridColumnStyle column);  
[C++] public: void Remove(DataGridColumnStyle* column);  
[VB] Public Sub Remove(ByVal column As DataGridColumnStyle)  
[JScript] public function Remove(column : DataGridColumnStyle); Removes the  
specified System.Windows.Forms.DataGridColumnStyle from the  
System.Windows.Forms.GridColumnStylesCollection .
```

*Description*

Removes the specified **System.Windows.Forms.DataGridColumnStyle** from the **System.Windows.Forms.GridColumnStylesCollection** .

Use the **System.Windows.Forms.GridColumnStylesCollection.Contains(System.ComponentModel.PropertyDescriptor)** method to determine whether the **System.Windows.Forms.DataGridColumnStyle** exists in the collection. The **System.Windows.Forms.DataGridColumnStyle** to remove from the collection.

*x) RemoveAt*

[C#]            public            void            RemoveAt(int            index);

[C++]            public:            void            RemoveAt(int            index);

[VB]            Public            Sub            RemoveAt(ByVal            index            As            Integer)

[JScript]            public            function            RemoveAt(index            :            int);

*Description*

Removes the **System.Windows.Forms.DataGridColumnStyle** with the specified index from the **System.Windows.Forms.GridColumnStylesCollection** .

Use the **System.Windows.Forms.GridColumnStylesCollection.Contains(System.ComponentModel.PropertyDescriptor)** method to determine whether the **System.Windows.Forms.DataGridColumnStyle** exists in the collection. The zero-based index of the **System.Windows.Forms.DataGridColumnStyle** to remove.

*y) ResetPropertyDescriptors*

[C#]            public            void            ResetPropertyDescriptors();

[C++]            public:            void            ResetPropertyDescriptors();

[VB]            Public            Sub            ResetPropertyDescriptors()

[JScript]            public            function            ResetPropertyDescriptors();



## Description

Sets the **System.ComponentModel.PropertyDescriptor** for each column style in the collection to **null**.

### z) *ICollection.CopyTo*

[C#] void ICollection.CopyTo(Array array, int index);

[C++] void ICollection::CopyTo(Array\* array, int index);

[VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements ICollection.CopyTo

[JScript] function ICollection.CopyTo(array : Array, index : int);

### aa) *IEnumerable.GetEnumerator*

[C#] IEnumerator IEnumerable.GetEnumerator();

[C++] IEnumerator\* IEnumerable::GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator Implements IEnumerable.GetEnumerator

[JScript] function IEnumerable.GetEnumerator() : IEnumerator;

### bb) *IList.Add*

[C#] int IList.Add(object value);

[C++] int IList::Add(Object\* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

cc) *IList.Clear*

[C#] void IList.Clear();  
[C++] void IList::Clear();  
[VB] Sub Clear() Implements IList.Clear  
[JScript] function IList.Clear();

dd) *IList.Contains*

[C#] bool IList.Contains(object value);  
[C++] bool IList::Contains(Object\* value);  
[VB] Function Contains(ByVal value As Object) As Boolean Implements  
IList.Contains  
[JScript] function IList.Contains(value : Object) : Boolean;

ee) *IList.IndexOf*

[C#] int IList.IndexOf(object value);  
[C++] int IList::IndexOf(Object\* value);  
[VB] Function IndexOf(ByVal value As Object) As Integer Implements  
IList.IndexOf  
[JScript] function IList.IndexOf(value : Object) : int;

ff) *IList.Insert*

[C#] void IList.Insert(int index, object value);  
[C++] void IList::Insert(int index, Object\* value);  
[VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

1 IList.Insert

2 [JScript] function IList.Insert(index : int, value : Object);

3 *gg) IList.Remove*

4  
5 [C#] void IList.Remove(object value);

6 [C++] void IList::Remove(Object\* value);

7 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

8 [JScript] function IList.Remove(value : Object);

9 *hh) IList.RemoveAt*

10  
11 [C#] void IList.RemoveAt(int index);

12 [C++] void IList::RemoveAt(int index);

13 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

14 [JScript] function IList.RemoveAt(index : int);

15 GridItem class (System.Windows.Forms)

16 *a) ToString*

17  
18  
19 *Description*

20 Implements one row in a **System.Windows.Forms.PropertyGrid** .

21 Grid items represent the hierarchy of the view into a  
22 **System.Windows.Forms.PropertyGrid** . You can use a  
23 **System.Windows.Forms.GridItem** to obtain information about the grid's  
state and contents.

24 *b) GridItem*

25 *Example Syntax:*

c) *ToString*

```
[C#]                protected                GridItem();
[C++]                protected:                GridItem();
[VB]                Protected                Sub                New()
[JScript] protected function GridItem();
```

d) *Expandable*

e) *ToString*

```
[C#]    public    virtual    bool    Expandable    {get;}
[C++]    public:    __property    virtual    bool    get_Expandable();
[VB]    Overridable    Public    ReadOnly    Property    Expandable    As    Boolean
[JScript]    public    function    get    Expandable()    :    Boolean;
```

*Description*

When overridden in a derived class, gets a value indicating whether the specified property is expandable.

f) *Expanded*

g) *ToString*

```
[C#]    public    virtual    bool    Expanded    {get;    set;}
[C++]    public:    __property    virtual    bool    get_Expanded();public:    __property    virtual
void                                            set_Expanded(bool);
[VB]    Overridable    Public    Property    Expanded    As    Boolean
[JScript]    public    function    get    Expanded()    :    Boolean;public    function    set
```

Expanded(Boolean);

*Description*

When overridden in a derived class, gets or sets a value indicating whether the **System.Windows.Forms.GridItem** is in an expanded state.

*h) GridItems*

*i) ToString*

[C#] public abstract GridItemCollection GridItems {get;}

[C++] public: \_\_property virtual GridItemCollection\* get\_GridItems() = 0;

[VB] MustOverride Public ReadOnly Property GridItems As GridItemCollection

[JScript] public abstract function get GridItems() : GridItemCollection;

*Description*

When overridden in a derived class, gets the collection of **System.Windows.Forms.GridItem** objects, if any, associated as a child of this **System.Windows.Forms.GridItem**.

*j) GridItemType*

*k) ToString*

[C#] public abstract GridItemType GridItemType {get;}

[C++] public: \_\_property virtual GridItemType get\_GridItemType() = 0;

[VB] MustOverride Public ReadOnly Property GridItemType As GridItemType

[JScript] public abstract function get GridItemType() : GridItemType;

*Description*

When overridden in a derived class, gets the type of this **System.Windows.Forms.GridItem** .

For a **System.Windows.Forms.GridItem** of type **System.Windows.Forms.GridItemType.Property** , you must also ensure that the **System.Windows.Forms.GridItem.PropertyDescriptor** has a valid value. For a **System.Windows.Forms.GridItem** of type **System.Windows.Forms.GridItemType.Root** the **System.Windows.Forms.GridItem.Parent** property must be **null** .

l) *Label*

m) *ToString*

[C#]            public            abstract            string            Label            {get;}

[C++]    public:    \_\_property    virtual    String\*    get\_Label()    =    0;

[VB]    MustOverride    Public    ReadOnly    Property    Label    As    String

[JScript]    public    abstract    function    get    Label()    :    String;

#### *Description*

When overridden in a derived class, gets the text of this **System.Windows.Forms.GridItem** .

This class gets the text that displays in the left column of the grid. The text retrieved can be different from the actual property name of the property represented by this **System.Windows.Forms.GridItem** . You can get the name for a **System.Windows.Forms.GridItem** of type **System.Windows.Forms.GridItemType.Property** by retrieving the **System.Windows.Forms.GridItem.PropertyDescriptor** and checking its **System.ComponentModel.MemberDescriptor.Name** property.

n) *Parent*

o) *ToString*

[C#]            public            abstract            GridItem            Parent            {get;}

```

1 [C++] public: __property virtual GridItem* get_Parent() = 0;
2 [VB] MustOverride Public ReadOnly Property Parent As GridItem
3 [JScript] public abstract function get Parent() : GridItem;

```

#### *Description*

When overridden in a derived class, gets the parent **System.Windows.Forms.GridItem** of this **System.Windows.Forms.GridItem** , if any.

*p) PropertyDescriptor*

*q) ToString*

```

11 [C#] public abstract PropertyDescriptor PropertyDescriptor {get;}
12 [C++] public: __property virtual PropertyDescriptor* get_PropertyDescriptor() =
13 0;
14 [VB] MustOverride Public ReadOnly Property PropertyDescriptor As
15 PropertyDescriptor
16 [JScript] public abstract function get PropertyDescriptor() : PropertyDescriptor;

```

#### *Description*

When overridden in a derived class, gets the **System.ComponentModel.PropertyDescriptor** that is associated with this **System.Windows.Forms.GridItem** .

This property is only valid for a **System.Windows.Forms.GridItem** of type **System.Windows.Forms.GridItemType.Property** .

- r) *Value*
- s) *ToString*

```
[C#]      public      abstract      object      Value      {get;}
[C++]    public:    __property    virtual    Object*    get_Value()    =    0;
[VB]     MustOverride    Public    ReadOnly    Property    Value    As    Object
[JScript]    public    abstract    function    get    Value()    :    Object;
```

#### *Description*

When overridden in a derived class, gets the current value of this **System.Windows.Forms.GridItem** .

- t) *Select*

```
[C#]      public      abstract      bool      Select();
[C++]    public:      virtual      bool      Select()      =      0;
[VB]     MustOverride    Public    Function    Select()    As    Boolean
[JScript]    public    abstract    function    Select()    :    Boolean;
```

#### *Description*

When overridden in a derived class, selects this **System.Windows.Forms.GridItem** in the **System.Windows.Forms.PropertyGrid** .

*Return Value:* **true** if the selection is successful; otherwise, **false** .



GridItemCollection class (System.Windows.Forms)

*a) ToString*

*Description*

Contains a collection of **System.Windows.Forms.GridItem** objects.

This class represents a collection of **System.Windows.Forms.GridItem** objects stored in a **System.Windows.Forms.PropertyGrid** .

*b) ToString*

```
[C#]      public      static      GridItemCollection      Empty;
[C++]     public:      static      GridItemCollection*      Empty;
[VB]      Public      Shared      Empty      As      GridItemCollection
[JScript] public      static      var      Empty      :      GridItemCollection;
```

*Description*

*c) Count*

*d) ToString*

```
[C#]      public      int      Count      {get;}
[C++]     public:      __property      int      get_Count();
[VB]      Public      ReadOnly      Property      Count      As      Integer
[JScript] public      function      get      Count()      :      int;
```

*Description*

Gets the number of grid items in the collection.

*e) Item*

*f) ToString*

[C#] public GridItem this[string label] {get;}

[C++] public: \_\_property GridItem\* get\_Item(String\* label);

[VB] Public Default ReadOnly Property Item(ByVal label As String) As GridItem

[JScript] returnValue = GridItemCollectionObject.Item(label);

*Description*

Gets the **System.Windows.Forms.GridItem** with the matching label. A string value to match to a grid item label

*g) Item*

*h) ToString*

[C#] public GridItem this[int index] {get;}

[C++] public: \_\_property GridItem\* get\_Item(int index);

[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As GridItem

[JScript] returnValue = GridItemCollectionObject.Item(index); Gets a **System.Windows.Forms.GridItem** from the collection.

*Description*

Gets the **System.Windows.Forms.GridItem** at the specified index. The index of the grid item to return.

*i) GetEnumerator*

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]        public:          __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript]    public    function    GetEnumerator()    :    IEnumerator;
```

*Description*

Returns an enumeration of all the grid items in the collection.  
*Return Value:* An **System.Collections.IEnumerator** for the **System.Windows.Forms.GridItemCollection** .

*j) ICollection.CopyTo*

```
[C#]    void    ICollection.CopyTo(Array    dest,    int    index);
[C++]    void    ICollection::CopyTo(Array*    dest,    int    index);
[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements
ICollection.CopyTo
[JScript] function ICollection.CopyTo(dest : Array, index : int);
```

GridItemType enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the valid grid item types for a **System.Windows.Forms.PropertyGrid** .

*b) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | GridItemType | ArrayValue;     |
| [C++]     | public: | const | GridItemType | ArrayValue;     |
| [VB]      | Public  | Const | ArrayValue   | As GridItemType |
| [JScript] | public  | var   | ArrayValue   | : GridItemType; |

*Description*

The **System.Windows.Forms.GridItem** is an element of an array.

*c) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | GridItemType | Category;       |
| [C++]     | public: | const | GridItemType | Category;       |
| [VB]      | Public  | Const | Category     | As GridItemType |
| [JScript] | public  | var   | Category     | : GridItemType; |

*Description*

A grid entry that is a category name. A category is a descriptive grouping for groups of **System.Windows.Forms.GridItem** rows. Typical categories include the following Behavior, Layout, Data, and Appearance.

*d) ToString*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | GridItemType | Property;       |
| [C++]     | public: | const | GridItemType | Property;       |
| [VB]      | Public  | Const | Property     | As GridItemType |
| [JScript] | public  | var   | Property     | : GridItemType; |

*Description*

A grid entry that corresponds to a property.

*e) ToString*

[C#]            public            const            GridItemType            Root;

[C++]            public:            const            GridItemType            Root;

[VB]            Public            Const            Root            As            GridItemType

[JScript]            public            var            Root            :            GridItemType;

*Description*

A root item in the grid hierarchy.

GridTablesFactory class (System.Windows.Forms)

*a) ToString*

*Description*

*b) CreateGridTables*

[C#] public static DataGridTableStyle[] CreateGridTables(DataGridTableStyle  
gridTable, object dataSource, string dataMember, BindingContext  
bindingManager);

[C++] public: static DataGridTableStyle\* CreateGridTables(DataGridTableStyle\*  
gridTable, Object\* dataSource, String\* dataMember, BindingContext\*

```

1 bindingManager)                                     [];
2 [VB] Public Shared Function CreateGridTables(ByVal gridTable As
3 DataGridTableStyle, ByVal dataSource As Object, ByVal dataMember As String,
4 ByVal bindingManager As BindingContext) As DataGridTableStyle()
5 [JScript] public static function CreateGridTables(gridTable : DataGridTableStyle,
6 dataSource : Object, dataMember : String, bindingManager : BindingContext) :
7 DataGridTableStyle[];

```

### *Description*

Takes a DataView and creates an intelligent mapping of DataView storage types into available DataColumn types.

*Return Value:* A GridTables containing DataGridTable objects which map to the DataTables "reachable" from the cursor specified. Only DataTables that are accessible from the DataTable associated with cursor through child relations will be contained in the collection.

GridTableStylesCollection class (System.Windows.Forms)

#### *a) ToString*

### *Description*

Represents a collection of **System.Windows.Forms.DataGridTableStyle** objects in the **System.Windows.Forms.DataGrid** control.

The **System.Windows.Forms.GridTableStylesCollection** contains **System.Windows.Forms.DataGridTableStyle** objects that allows the **System.Windows.Forms.DataGrid** control to display a customized grid style for each **System.Data.DataTable** in a **System.Data.DataSet**.

1       **b)     Count**

2       **c)     IsReadOnly**

3       **d)     IsSynchronized**

4       **e)     Item**

5       **f)     ToString**

6       **System.Windows.Forms.DataGridTableStyle**

7  
8       *Description*

9       Gets the **System.Windows.Forms.DataGridTableStyle** specified by index.

10      Use the

11      **System.Windows.Forms.GridTableStylesCollection.Add(System.Windo**  
12      **ws.Forms.DataGridTableStyle)** or  
13      **System.Windows.Forms.GridTableStylesCollection.AddRange(System.**  
14      **Windows.Forms.DataGridTableStyle[])** method to add items to the  
15      collection. The index of the **System.Windows.Forms.DataGridTableStyle** to  
16      get.

14       **g)     Item**

15       **h)     ToString**

17      [C#]     public     DataGridTableStyle     this[string     tableName]     {get;}  
18

19      [C++]    public: \_\_property DataGridTableStyle\* get\_Item(String\*   tableName);  
20

21      [VB]    Public Default ReadOnly Property Item(ByVal tableName As String) As  
22      DataGridTableStyle  
23

24      [JScript]   returnValue   =   GridTableStylesCollectionObject.Item(tableName);  
25

23       *Description*

24       Gets the **System.Windows.Forms.DataGridTableStyle** with the specified  
25       name.





[VB] Overridable Public Function Add(ByVal table As DataGridTableStyle) As Integer

[JScript] public function Add(table : DataGridTableStyle) : int;

*Description*

Adds a **System.Windows.Forms.DataGridTableStyle** to this collection.

*Return Value:* The index of the newly added object. The

**System.Windows.Forms.DataGridTableStyle** to add to the collection.

*n) AddRange*

[C#] public virtual void AddRange(DataGridTableStyle[] tables);

[C++] public: virtual void AddRange(DataGridTableStyle\* tables[]);

[VB] Overridable Public Sub AddRange(ByVal tables() As DataGridTableStyle)

[JScript] public function AddRange(tables : DataGridTableStyle[]);

*Description*

Adds an array of table styles to the collection. An array of **System.Windows.Forms.DataGridTableStyle** objects.

*o) Clear*

[C#] public void Clear();

[C++] public: void Clear();

[VB] Public Sub Clear()

[JScript] public function Clear();

*Description*

Clears the collection.

*p) Contains*

[C#]        public        bool        Contains(DataGridTableStyle        table);  
[C++]       public:        bool        Contains(DataGridTableStyle\*        table);  
[VB] Public Function Contains(ByVal table As DataGridTableStyle) As Boolean  
[JScript] public function Contains(table : DataGridTableStyle) : Boolean; Gets a  
value                    indicating                    whether                    the  
**System.Windows.Forms.GridTableStylesCollection** contains the specified  
**System.Windows.Forms.DataGridTableStyle** .

*Description*

Gets a value indicating whether the  
**System.Windows.Forms.GridTableStylesCollection** contains the specified  
**System.Windows.Forms.DataGridTableStyle** .

*Return Value:* **true** if the specified table style exists in the collection; otherwise,  
**false** . The **System.Windows.Forms.DataGridTableStyle** to look for.

*q) Contains*

[C#]        public        bool        Contains(string        name);  
[C++]       public:        bool        Contains(String\*        name);  
[VB] Public Function Contains(ByVal name As String) As Boolean  
[JScript] public function Contains(name : String) : Boolean;

*Description*

Gets a value indicating whether the  
**System.Windows.Forms.GridTableStylesCollection** contains the  
**System.Windows.Forms.DataGridTableStyle** specified by name.

*Return Value:* **true** if the specified table style exists in the collection; otherwise, **false** . The **System.Windows.Forms.DataGridTableStyle.MappingName** of the **System.Windows.Forms.DataGridTableStyle** to look for.

*r) OnCollectionChanged*

[C#] protected void OnCollectionChanged(CollectionChangeEventArgs ccevent);

[C++] protected: void OnCollectionChanged(CollectionChangeEventArgs\* ccevent);

[VB] Protected Sub OnCollectionChanged(ByVal ccevent As CollectionChangeEventArgs)

[JScript] protected function OnCollectionChanged(ccevent : CollectionChangeEventArgs);

*Description*

Raises the **System.Windows.Forms.GridTableStylesCollection.CollectionChanged** event. A **System.ComponentModel.CollectionChangeEventArgs** containing the event data.

*s) Remove*

[C#] public void Remove(DataGridTableStyle table);

[C++] public: void Remove(DataGridTableStyle\* table);

[VB] Public Sub Remove(ByVal table As DataGridTableStyle)

[JScript] public function Remove(table : DataGridTableStyle);

*Description*

Removes the specified **System.Windows.Forms.DataGridTableStyle** .

Use the

**System.Windows.Forms.GridTableStylesCollection.Contains(System.Windows.Forms.DataGridTableStyle)** method to determine if a specific **System.Windows.Forms.DataGridTableStyle** object exists before using the **System.Windows.Forms.GridTableStylesCollection.Remove(System.Windows.Forms.DataGridTableStyle)** method. The **System.Windows.Forms.DataGridTableStyle** to remove.

*t) RemoveAt*

|           |         |          |                      |             |
|-----------|---------|----------|----------------------|-------------|
| [C#]      | public  | void     | RemoveAt(int         | index);     |
| [C++]     | public: | void     | RemoveAt(int         | index);     |
| [VB]      | Public  | Sub      | RemoveAt(ByVal index | As Integer) |
| [JScript] | public  | function | RemoveAt(index       | : int);     |

*Description*

Removes a **System.Windows.Forms.DataGridTableStyle** at the specified index.

Use the

**System.Windows.Forms.GridTableStylesCollection.Contains(System.Windows.Forms.DataGridTableStyle)** method to determine if a specific **System.Windows.Forms.DataGridTableStyle** object exists before using the **System.Windows.Forms.GridTableStylesCollection.Remove(System.Windows.Forms.DataGridTableStyle)** method. The index of the **System.Windows.Forms.DataGridTableStyle** to remove.

*u) ICollection.CopyTo*

|           |                    |                              |                         |            |         |
|-----------|--------------------|------------------------------|-------------------------|------------|---------|
| [C#]      | void               | ICollection.CopyTo(Array     | array,                  | int        | index); |
| [C++]     | void               | ICollection::CopyTo(Array*   | array,                  | int        | index); |
| [VB]      | Sub                | CopyTo(ByVal array As Array, | ByVal index As Integer) | Implements |         |
|           | ICollection.CopyTo |                              |                         |            |         |
| [JScript] | function           | ICollection.CopyTo(array     | : Array,                | index      | : int); |

v) ***IEnumerable.GetEnumerator***

[C#]                   IEnumerator                   IEnumerable.GetEnumerator();  
 [C++]                   IEnumerator\*                   IEnumerable::GetEnumerator();  
 [VB]    Function    GetEnumerator()    As    IEnumerator    Implements  
 IEnumerable.GetEnumerator  
 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;

w) ***IList.Add***

[C#]                   int                   IList.Add(object                   value);  
 [C++]                   int                   IList::Add(Object\*                   value);  
 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add  
 [JScript] function IList.Add(value : Object) : int;

x) ***IList.Clear***

[C#]                                   void                                   IList.Clear();  
 [C++]                                   void                                   IList::Clear();  
 [VB]                   Sub                   Clear()                   Implements                   IList.Clear  
 [JScript] function IList.Clear();

y) ***IList.Contains***

[C#]                   bool                   IList.Contains(object                   value);  
 [C++]                   bool                   IList::Contains(Object\*                   value);  
 [VB] Function Contains(ByVal value As Object) As Boolean Implements

1 IList.Contains

2 [JScript] function IList.Contains(value : Object) : Boolean;

3       **z)     IList.IndexOf**

4  
5 [C#]               int               IList.IndexOf(object               value);

6 [C++]               int               IList::IndexOf(Object\*               value);

7 [VB] Function IndexOf(ByVal value As Object) As Integer Implements

8 IList.IndexOf

9 [JScript] function IList.IndexOf(value : Object) : int;

10       **aa)    IList.Insert**

11  
12 [C#]       void       IList.Insert(int       index,       object       value);

13 [C++]       void       IList::Insert(int       index,       Object\*       value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17       **bb)    IList.Remove**

18  
19 [C#]               void               IList.Remove(object               value);

20 [C++]               void               IList::Remove(Object\*               value);

21 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

22 [JScript] function IList.Remove(value : Object);

cc) *IList.RemoveAt*

[C#] void IList.RemoveAt(int index);

[C++] void IList::RemoveAt(int index);

[VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt

[JScript] function IList.RemoveAt(index : int);

GroupBox class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a Windows group box.

The **System.Windows.Forms.GroupBox** displays a frame around a group of controls with or without a caption. Use a **System.Windows.Forms.GroupBox** to logically group a collection of controls on a form. The group box is a container control that can be used to define groups of controls.

b) *GroupBox*

*Example Syntax:*

c) *ToString*

[C#] public GroupBox();

[C++] public: GroupBox();

[VB] Public Sub New()

[JScript] public function GroupBox();

*Description*

1 Initializes a new instance of the **System.Windows.Forms.GroupBox** class.

2 By default, the **System.Windows.Forms.Control.TabStop** property is set to  
3 **false** when a new **System.Windows.Forms.GroupBox** is created. The group  
4 box has an initial height of 100 pixels and width of 200 pixels.

- 5 *d) AccessibilityObject*
- 6 *e) AccessibleDefaultActionDescription*
- 7 *f) AccessibleDescription*
- 8 *g) AccessibleName*
- 9 *h) AccessibleRole*
- 10 *i) AllowDrop*
- 11 *j) ToString*

12  
13 *Description*

14 Gets or sets a value indicating whether the control will allow drag and drop  
15 operations and events to be used.  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25



|    |            |                                |
|----|------------|--------------------------------|
| 1  | <b>k)</b>  | <b><i>Anchor</i></b>           |
| 2  | <b>l)</b>  | <b><i>BackColor</i></b>        |
| 3  | <b>m)</b>  | <b><i>BackgroundImage</i></b>  |
| 4  | <b>n)</b>  | <b><i>BindingContext</i></b>   |
| 5  | <b>o)</b>  | <b><i>Bottom</i></b>           |
| 6  | <b>p)</b>  | <b><i>Bounds</i></b>           |
| 7  | <b>q)</b>  | <b><i>CanFocus</i></b>         |
| 8  | <b>r)</b>  | <b><i>CanSelect</i></b>        |
| 9  | <b>s)</b>  | <b><i>Capture</i></b>          |
| 10 | <b>t)</b>  | <b><i>CausesValidation</i></b> |
| 11 | <b>u)</b>  | <b><i>ClientRectangle</i></b>  |
| 12 | <b>v)</b>  | <b><i>ClientSize</i></b>       |
| 13 | <b>w)</b>  | <b><i>CompanyName</i></b>      |
| 14 | <b>x)</b>  | <b><i>Container</i></b>        |
| 15 | <b>y)</b>  | <b><i>ContainsFocus</i></b>    |
| 16 | <b>z)</b>  | <b><i>ContextMenu</i></b>      |
| 17 | <b>aa)</b> | <b><i>Controls</i></b>         |
| 18 | <b>bb)</b> | <b><i>Created</i></b>          |
| 19 | <b>cc)</b> | <b><i>CreateParams</i></b>     |
| 20 | <b>dd)</b> | <b><i>ToString</i></b>         |
| 21 |            |                                |
| 22 |            |                                |
| 23 |            |                                |
| 24 |            | <b><i>Description</i></b>      |
| 25 |            |                                |

1  
2 *ee) Cursor*

3 *ff) DataBindings*

4 *gg) DefaultImeMode*

5 *hh) DefaultSize*

6 *ii) ToString*

7  
8  
9 *Description*

10 Deriving classes can override this to configure a default size for their control.  
11 This is more efficient than setting the size in the control's constructor.

12 *jj) DesignMode*

13 *kk) DisplayRectangle*

14 *ll) ToString*

15  
16  
17 *Description*

18 Gets a rectangle that represents the dimensions of the  
19 **System.Windows.Forms.GroupBox** .

20 The rectangle is Gets a rectangle that represents the dimensions of the  
21 **System.Windows.Forms.GroupBox** .

*mm) Disposing*

*nn) Dock*

*oo) Enabled*

*pp) Events*

*qq) FlatStyle*

*rr) ToString*

*Description*

Gets or sets the flat style appearance of the group box control.

ss) *Focused*  
 tt) *Font*  
 uu) *FontHeight*  
 vv) *ForeColor*  
 ww) *Handle*  
 xx) *HasChildren*  
 yy) *Height*  
 zz) *ImeMode*  
 aaa) *InvokeRequired*  
 bbb) *IsAccessible*  
 ccc) *IsDisposed*  
 ddd) *IsHandleCreated*  
 eee) *Left*  
 fff) *Location*  
 ggg) *Name*  
 hhh) *Parent*  
 iii) *ProductName*  
 jjj) *ProductVersion*  
 kkk) *RecreatingHandle*  
 ll) *Region*  
 mmm) *RenderRightToLeft*  
 nnn) *ResizeRedraw*  
 ooo) *Right*

1      *ppp) RightToLeft*

2      *qqq) ShowFocusCues*

3      *rrr) ShowKeyboardCues*

4      *sss) Site*

5      *ttt) Size*

6      *uuu) TabIndex*

7      *vvv) TabStop*

8      *www) ToString*

9  
10  
11      *Description*

12      Gets or sets a value indicating whether the user may press the TAB key to give  
13      the focus to the **System.Windows.Forms.GroupBox** .

14      *xxx) Tag*

15      *yyy) Text*

16      *zzz) ToString*

17  
18  
19      *Description*

aaaa) *Top*

bbbb) *TopLevelControl*

cccc) *Visible*

dddd) *Width*

eeee) *WindowTarget*

ffff) *ToString*

gggg) *ToString*

hhhh) *ToString*

iiii) *ToString*

jjjj) *ToString*

kkkk) *ToString*

llll) *ToString*

mmmm) *ToString*

nnnn) *ToString*

oooo) *ToString*

pppp) *OnFontChanged*

[C#]     protected     override     void     OnFontChanged(EventArgs     e);

[C++]     protected:     void     OnFontChanged(EventArgs\*     e);

[VB]     Overrides     Protected     Sub     OnFontChanged(ByVal     e     As     EventArgs)

[JScript]     protected     override     function     OnFontChanged(e     :     EventArgs);

*Description*

### *qqqq) OnPaint*

```
1
2 [C#]      protected      override      void      OnPaint(PaintEventArgs      e);
3
4 [C++]      protected:      void      OnPaint(PaintEventArgs*      e);
5
6 [VB] Overrides Protected Sub OnPaint(ByVal e As PaintEventArgs)
7
8 [JScript] protected override function OnPaint(e : PaintEventArgs);
```

### *rrrr) ProcessMnemonic*

```
9
10 [C#]      protected      override      bool      ProcessMnemonic(char      charCode);
11
12 [C++]      protected:      bool      ProcessMnemonic(__wchar_t      charCode);
13
14 [VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)
15
16 As Boolean
17
18 [JScript] protected override function ProcessMnemonic(charCode : Char) :
19
20 Boolean;
```

### *Description*

We use this to process mnemonics and send them on to the first child control.

### *ssss) ToString*

```
21
22 [C#]      public      override      string      ToString();
23
24 [C++]      public:      String*      ToString();
25
26 [VB] Overrides Public Function ToString() As String
27
28 [JScript] public override function ToString() : String;
```

### *Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

*tttt) WndProc*

[C#] protected override void WndProc(ref Message m);

[C++] protected: void WndProc(Message\* m);

[VB] Overrides Protected Sub WndProc(ByRef m As Message)

[JScript] protected override function WndProc(m : Message);

### *Description*

Help class (System.Windows.Forms)

*a) WndProc*

### *Description*

Encapsulates the HTML Help 1.0 engine.

You cannot create a new instance of the **System.Windows.Forms.Help** class. To provide Help to an application, call the static

**System.Windows.Forms.Help.ShowHelp(System.Windows.Forms.Control, System.String)** and **System.Windows.Forms.Help.ShowHelpIndex(System.Windows.Forms.Control, System.String)** methods.

*b) ShowHelp*

[C#] public static void ShowHelp(Control parent, string url);

[C++] public: static void ShowHelp(Control\* parent, String\* url);



[VB] Public Shared Sub ShowHelp(ByVal parent As Control, ByVal url As String)

[JScript] public static function ShowHelp(parent : Control, url : String); Displays the contents of a Help file.

### *Description*

Displays the contents of the Help file at the specified URL.

The *url* parameter can be of the form C:\path\sample.chm or /folder/file.htm. A **System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The path and name of the Help file.

### *c) ShowHelp*

[C#] public static void ShowHelp(Control parent, string url, HelpNavigator navigator);

[C++] public: static void ShowHelp(Control\* parent, String\* url, HelpNavigator navigator);

[VB] Public Shared Sub ShowHelp(ByVal parent As Control, ByVal url As String, ByVal navigator As HelpNavigator)

[JScript] public static function ShowHelp(parent : Control, url : String, navigator : HelpNavigator);

### *Description*

Displays the contents of the Help file found at the specified URL for a specific topic.

The *url* parameter can be of the form C:\path\sample.chm or /folder/file.htm. A **System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The path and name of the Help file. One of the **System.Windows.Forms.HelpNavigator** values.

#### d) *ShowHelp*

```
1 [C#] public static void ShowHelp(Control parent, string url, string keyword);
2
3 [C++] public: static void ShowHelp(Control* parent, String* url, String*
4 keyword);
5
6 [VB] Public Shared Sub ShowHelp(ByVal parent As Control, ByVal url As
7 String, ByVal keyword As String)
8
9 [JScript] public static function ShowHelp(parent : Control, url : String, keyword :
10 String);
```

#### *Description*

Displays the contents of the Help file found at the specified URL for a specific keyword.

The *url* parameter can be of the form C:\path\sample.chm or /folder/file.htm. A **System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The path and name of the Help file. The keyword to display Help for.

#### e) *ShowHelp*

```
17 [C#] public static void ShowHelp(Control parent, string url, HelpNavigator
18 command, object param);
19
20 [C++] public: static void ShowHelp(Control* parent, String* url, HelpNavigator
21 command, Object* param);
22
23 [VB] Public Shared Sub ShowHelp(ByVal parent As Control, ByVal url As
24 String, ByVal command As HelpNavigator, ByVal param As Object)
25
26 [JScript] public static function ShowHelp(parent : Control, url : String, command :
27 HelpNavigator, param : Object);
```

## Description

Displays the contents of the Help file located at the URL supplied by the user.

Compiled help files provide table of contents, index, search, and keyword links in pages. You can use the following values for *command* :

**System.Windows.Forms.HelpNavigator.TableOfContents** ,

**System.Windows.Forms.HelpNavigator.Find** ,

**System.Windows.Forms.HelpNavigator.Index** , or

**System.Windows.Forms.HelpNavigator.Topic** . A

**System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The path and name of the Help file. One of the

**System.Windows.Forms.HelpNavigator** values. The numeric id of the topic to display.

### f) *ShowHelpIndex*

[C#] public static void ShowHelpIndex(Control parent, string url);

[C++] public: static void ShowHelpIndex(Control\* parent, String\* url);

[VB] Public Shared Sub ShowHelpIndex(ByVal parent As Control, ByVal url As String)

[JScript] public static function ShowHelpIndex(parent : Control, url : String);

## Description

Displays the index of the specified Help file.

The *url* parameter can be of the form C:\path\sample.chm or /folder/file.htm. A **System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The path and name of the Help file.

### g) *ShowPopup*

[C#] public static void ShowPopup(Control parent, string caption, Point location);

[C++] public: static void ShowPopup(Control\* parent, String\* caption, Point

```

1 location);
2 [VB] Public Shared Sub ShowPopup(ByVal parent As Control, ByVal caption As
3 String,          ByVal          location          As          Point)
4 [JScript] public static function ShowPopup(parent : Control, caption : String,
5 location          :          Point);
6

```

### *Description*

Displays a help pop-up window. A **System.Windows.Forms.Control** that identifies the parent of the Help dialog box. The message to display in the pop-up window. A value specifying the horizontal and vertical coordinates at which to display the pop-up window.

HelpEventArgs class (System.Windows.Forms)

#### *a) ToString*

### *Description*

Provides data for the **System.Windows.Forms.Control.HelpRequested** event.

The **System.Windows.Forms.Control.HelpRequested** event occurs when the user requests help for a control. A

**System.Windows.Forms.HelpEventArgs** object specifies the screen coordinates of the mouse pointer and whether the event was handled.

#### *b) HelpEventArgs*

#### *Example Syntax:*

#### *c) ToString*

```

24 [C#]          public          HelpEventArgs(Point          mousePos);
25 [C++]         public:         HelpEventArgs(Point          mousePos);

```

```

1  [VB]      Public      Sub      New(ByVal      mousePos      As      Point)
2  [JScript]  public      function      HelpEventArgs(mousePos      :      Point);
3

```

#### *Description*

Initializes a new instance of the **System.Windows.Forms.HelpEventArgs** class. The coordinates of the mouse pointer.

d) *Handled*

e) *ToString*

```

9  [C#]      public      bool      Handled      {get;      set;}
10

```

```

11 [C++]  public:  __property  bool  get_Handled();public:  __property  void
12 set_Handled(bool);

```

```

13 [VB]      Public      Property      Handled      As      Boolean

```

```

14 [JScript]  public  function  get  Handled()  :  Boolean;public  function  set
15 Handled(Boolean);

```

#### *Description*

Gets or sets a value indicating whether the help event was handled.

If you do not set this property to **true** the event will be passed to Windows for additional processing.

f) *MousePos*

g) *ToString*

```

23 [C#]      public      Point      MousePos      {get;}

```

```

24 [C++]  public:      __property      Point      get_MousePos();

```

|           |        |          |          |            |    |        |
|-----------|--------|----------|----------|------------|----|--------|
| [VB]      | Public | ReadOnly | Property | MousePos   | As | Point  |
| [JScript] | public | function | get      | MousePos() | :  | Point; |

*Description*

Gets the screen coordinates of the mouse pointer.

You can use this information to provide help based on the position of the mouse pointer.

HelpEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **System.Windows.Forms.Control.HelpRequested** event of a **System.Windows.Forms.Control**. The source of the event. A **System.Windows.Forms.HelpEventArgs** that contains the event data.

When you create a **System.Windows.Forms.HelpEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

HelpNavigator enumeration (System.Windows.Forms)

a) *ToString*

*Description*

Specifies constants indicating which elements of the Help file to display.

This enumeration is used by **System.Windows.Forms.Help** and **System.Windows.Forms.HelpProvider** to provide access to specific elements of the Help file.

*b) ToString*

|           |         |       |                |                  |
|-----------|---------|-------|----------------|------------------|
| [C#]      | public  | const | HelpNavigator  | AssociateIndex;  |
| [C++]     | public: | const | HelpNavigator  | AssociateIndex;  |
| [VB]      | Public  | Const | AssociateIndex | As HelpNavigator |
| [JScript] | public  | var   | AssociateIndex | : HelpNavigator; |

*Description*

Specifies that the index for a specified topic is performed in the specified URL.

*c) ToString*

|           |         |       |               |                  |
|-----------|---------|-------|---------------|------------------|
| [C#]      | public  | const | HelpNavigator | Find;            |
| [C++]     | public: | const | HelpNavigator | Find;            |
| [VB]      | Public  | Const | Find          | As HelpNavigator |
| [JScript] | public  | var   | Find          | : HelpNavigator; |

*Description*

Specifies that the search page of a specified URL is displayed.

*d) ToString*

|       |         |       |               |                  |
|-------|---------|-------|---------------|------------------|
| [C#]  | public  | const | HelpNavigator | Index;           |
| [C++] | public: | const | HelpNavigator | Index;           |
| [VB]  | Public  | Const | Index         | As HelpNavigator |

1 [JScript] public var Index : HelpNavigator;

2

3 *Description*

4 Specifies that the index of a specified URL is displayed.

5 *e) ToString*

6

7 [C#] public const HelpNavigator KeywordIndex;

8 [C++] public: const HelpNavigator KeywordIndex;

9 [VB] Public Const KeywordIndex As HelpNavigator

10 [JScript] public var KeywordIndex : HelpNavigator;

11

12 *Description*

13 Specifies a keyword to search for and the action to take in the specified URL.

14 *f) ToString*

15

16 [C#] public const HelpNavigator TableOfContents;

17 [C++] public: const HelpNavigator TableOfContents;

18 [VB] Public Const TableOfContents As HelpNavigator

19 [JScript] public var TableOfContents : HelpNavigator;

20

21 *Description*

22 Specifies that the table of contents of the HTML 1.0 Help file is displayed.

23 *g) ToString*

24

25 [C#] public const HelpNavigator Topic;



|   |           |         |       |               |                  |
|---|-----------|---------|-------|---------------|------------------|
| 1 | [C++]     | public: | const | HelpNavigator | Topic;           |
| 2 | [VB]      | Public  | Const | Topic         | As HelpNavigator |
| 3 | [JScript] | public  | var   | Topic         | : HelpNavigator; |

4

5 *Description*

6 Specifies that the topic referenced by the specified URL is displayed.

7 HelpProvider class (System.Windows.Forms)

8 *a) ToString*

9

10

11 *Description*

12 Provides pop-up or online Help for controls.

13 Each instance of **System.Windows.Forms.HelpProvider** maintains a  
 14 collection of references to controls associated with it. To associate a Help file  
 15 with the **System.Windows.Forms.HelpProvider** object, set the  
 16 **System.Windows.Forms.HelpProvider.HelpNamespace** property. You  
 17 specify the type of Help provided by calling  
 18 **System.Windows.Forms.HelpProvider.SetHelpNavigator(System.Windo**  
 19 **ws.Forms.Control,System.Windows.Forms.HelpNavigator)** and providing  
 20 a **System.Windows.Forms.HelpNavigator** value for the specified control.  
 21 You provide the keyword or topic for the Help by calling  
 22 **System.Windows.Forms.HelpProvider.SetHelpKeyword(System.Windo**  
 23 **ws.Forms.Control,System.String)** .

19 *b) HelpProvider*

20 *Example Syntax:*

21 *c) ToString*

|    |       |         |                 |
|----|-------|---------|-----------------|
| 23 | [C#]  | public  | HelpProvider(); |
| 24 | [C++] | public: | HelpProvider(); |

|           |        |          |                 |
|-----------|--------|----------|-----------------|
| [VB]      | Public | Sub      | New()           |
| [JScript] | public | function | HelpProvider(); |

#### *Description*

Initializes a new instance of the **System.Windows.Forms.HelpProvider** class.

Once you have created an instance of **System.Windows.Forms.HelpProvider**, use **System.Windows.Forms.HelpProvider.HelpNamespace**, **System.Windows.Forms.HelpProvider.SetHelpKeyword(System.Windows.Forms.Control,System.String)**, and **System.Windows.Forms.HelpProvider.SetHelpNavigator(System.Windows.Forms.Control,System.Windows.Forms.HelpNavigator)** to associate your help topics with a control.

- d) *Container*
- e) *DesignMode*
- f) *Events*
- g) *HelpNamespace*
- h) *ToString*

#### *Description*

Gets or sets a value specifying the name of the Help file associated with this **System.Windows.Forms.HelpProvider** object.

The file name retrieved by this method identifies the file that provides Help support for all the controls for which this object provides Help. This file name can designate either a compiled Help file (.chm) or a raw HTML file.

i) *Site*

j) *CanExtend*

```
[C#]      public      virtual      bool      CanExtend(object      target);  
[C++]     public:     virtual      bool      CanExtend(Object*      target);  
[VB] Overridable Public Function CanExtend(ByVal target As Object) As  
Boolean  
[JScript] public function CanExtend(target : Object) : Boolean;
```

*Description*

Specifies whether this object can provide its extender properties to the specified object. The object

k) *GetHelpKeyword*

```
[C#]      public      virtual      string      GetHelpKeyword(Control      ctl);  
[C++]     public:     virtual      String*     GetHelpKeyword(Control*      ctl);  
[VB] Overridable Public Function GetHelpKeyword(ByVal ctl As Control) As  
String  
[JScript] public function GetHelpKeyword(ctl : Control) : String;
```

*Description*

Gets the Help keyword for the specified control.

*Return Value:* The Help topic associated with this control. If the

**System.Windows.Forms.HelpProvider** object is currently configured to display the entire Help file or is configured to provide a Help string, this method returns **null** . A **System.Windows.Forms.Control** from which to retrieve the Help topic.

### l) *GetHelpNavigator*

```
[C#]    public    virtual    HelpNavigator    GetHelpNavigator(Control    ctl);  
[C++]   public:   virtual    HelpNavigator    GetHelpNavigator(Control*    ctl);  
[VB]    Overridable Public Function GetHelpNavigator(ByVal ctl As Control) As  
HelpNavigator  
[JScript] public function GetHelpNavigator(ctl : Control) : HelpNavigator;
```

#### *Description*

Gets the current **System.Windows.Forms.HelpNavigator** setting for the specified control.

*Return Value:* The **System.Windows.Forms.HelpNavigator** setting for the specified control. The default is **null** . A **System.Windows.Forms.Control** from which to retrieve the Help string.

### m) *GetHelpString*

```
[C#]    public    virtual    string    GetHelpString(Control    ctl);  
[C++]   public:   virtual    String*    GetHelpString(Control*    ctl);  
[VB]    Overridable Public Function GetHelpString(ByVal ctl As Control) As String  
[JScript] public function GetHelpString(ctl : Control) : String;
```

#### *Description*

Gets the contents of the pop-up Help window for the specified control.

*Return Value:* The Help string associated with this control. The default value is **null** .

To display this Help string at run time, the user presses the F1 Key while the specified control has the input focus. A **System.Windows.Forms.Control** from which to retrieve the Help string.

n) *GetShowHelp*

```
[C#]      public      virtual      bool      GetShowHelp(Control      ctl);
[C++]     public:     virtual      bool      GetShowHelp(Control*      ctl);
[VB]      Overridable Public Function GetShowHelp(ByVal ctl As Control) As
Boolean
[JScript] public      function      GetShowHelp(ctl : Control) : Boolean;
```

*Description*

Gets a value indicating whether the specified control is associated with this

**System.Windows.Forms.HelpProvider** .

*Return Value:* **true** if Help will be displayed for the control; otherwise, **false** .

When you use the

**System.Windows.Forms.HelpProvider.SetHelpKeyword(System.Windows.Forms.Control,System.String)** or

**System.Windows.Forms.HelpProvider.SetHelpString(System.Windows.Forms.Control,System.String)** method to associate a keyword or prompt with

a specified control, calling this method automatically returns **true** . You can override this behavior by passing **false** to the

**System.Windows.Forms.HelpProvider.SetShowHelp(System.Windows.Forms.Control,System.Boolean)** method. A

**System.Windows.Forms.Control** for which Help will be displayed.

o) *ResetShowHelp*

```
[C#]      public      virtual      void      ResetShowHelp(Control      ctl);
[C++]     public:     virtual      void      ResetShowHelp(Control*      ctl);
[VB]      Overridable Public Sub      ResetShowHelp(ByVal ctl As Control)
[JScript] public      function      ResetShowHelp(ctl : Control);
```

*Description*

Used by the designer Used by the designer

*p) SetHelpKeyword*

```
[C#] public virtual void SetHelpKeyword(Control ctl, string keyword);  
[C++] public: virtual void SetHelpKeyword(Control* ctl, String* keyword);  
[VB] Overridable Public Sub SetHelpKeyword(ByVal ctl As Control, ByVal  
keyword As String)  
[JScript] public function SetHelpKeyword(ctl : Control, keyword : String);
```

*Description*

Specifies the keyword used to retrieve Help when the user invokes Help for the specified control.

The Help keyword provides the key information to retrieve the help associated with this control from the Help file specified by **System.Windows.Forms.HelpProvider.HelpNamespace** . To clear the keyword call **System.Windows.Forms.HelpProvider.SetHelpKeyword(System.Windows.Forms.Control, System.String)** with a *keyword* value of **null** . A **System.Windows.Forms.Control** that specifies the control for which to set the Help topic. The Help keyword to associate with the control.

*q) SetHelpNavigator*

```
[C#] public virtual void SetHelpNavigator(Control ctl, HelpNavigator navigator);  
[C++] public: virtual void SetHelpNavigator(Control* ctl, HelpNavigator  
navigator);  
[VB] Overridable Public Sub SetHelpNavigator(ByVal ctl As Control, ByVal  
navigator As HelpNavigator)  
[JScript] public function SetHelpNavigator(ctl : Control, navigator :
```

HelpNavigator);

### *Description*

Specifies the Help command to use when retrieving Help from the Help file for the specified control. A **System.Windows.Forms.Control** for which to set the Help keyword. The Help keyword to associate with the control; or **null**, if the contents of the entire Help file display.

#### *r) SetHelpString*

[C#] public virtual void SetHelpString(Control ctl, string helpString);

[C++] public: virtual void SetHelpString(Control\* ctl, String\* helpString);

[VB] Overridable Public Sub SetHelpString(ByVal ctl As Control, ByVal helpString As String)

[JScript] public function SetHelpString(ctl : Control, helpString : String);

### *Description*

Specifies the Help string associated with the specified control.

The Help string that you specify in the *helpString* parameter is displayed in a pop-up window when the user presses the F1 Key while the specified control has focus. A **System.Windows.Forms.Control** with which to associate the Help string. The Help string associated with the control.

#### *s) SetShowHelp*

[C#] public virtual void SetShowHelp(Control ctl, bool value);

[C++] public: virtual void SetShowHelp(Control\* ctl, bool value);

[VB] Overridable Public Sub SetShowHelp(ByVal ctl As Control, ByVal value As Boolean)

1 [JScript] public function SetShowHelp(ctl : Control, value : Boolean);

3 *Description*

4 Specifies whether Help is displayed for the specified control.

5 If you previously called the

6 **System.Windows.Forms.HelpProvider.SetHelpString(System.Windows.Forms.Control, System.String)** or

7 **System.Windows.Forms.HelpProvider.SetHelpKeyword(System.Windows.Forms.Control, System.String)** for the control specified in the *ctl*

8 parameter, the **System.Windows.Forms.HelpProvider** object automatically displays Help for that control. To modify this behavior, call the

9 **System.Windows.Forms.HelpProvider.SetShowHelp(System.Windows.Forms.Control, System.Boolean)** method, specifying **false** in the *value*

10 parameter. A **System.Windows.Forms.Control** for which Help is turned on or off. **true** if Help displays for the control; otherwise, **false**.

11 t) *ToString*

12  
13 [C#] public override string ToString();

14 [C++] public: String\* ToString();

15 [VB] Overrides Public Function ToString() As String

16 [JScript] public override function ToString() : String;

18 *Description*

19 Returns a string representation for this control.

20 *Return Value:* String Returns a string representation for this control.

21 MonthCalendar.HitArea enumeration (System.Windows.Forms)

22 a) *ToString*

25 *Description*



Defines constants that represent areas in a  
**System.Windows.Forms.MonthCalendar** control.

This enumeration has specific areas of the  
**System.Windows.Forms.MonthCalendar** control as its enumerated values.  
The **System.Windows.Forms.MonthCalendar.HitTestInfo.HitArea**  
member of **System.Windows.Forms.MonthCalendar.HitTestInfo** will be  
one of these enumerated values, and indicates which portion of a month  
calendar is under a specified point.

*b) ToString*

[C#] public const MonthCalendar.HitArea CalendarBackground;

[C++] public: const MonthCalendar.HitArea CalendarBackground;

[VB] Public Const CalendarBackground As MonthCalendar.HitArea

[JScript] public var CalendarBackground : MonthCalendar.HitArea;

*Description*

The specified point is part of the calendar's background.

*c) ToString*

[C#] public const MonthCalendar.HitArea Date;

[C++] public: const MonthCalendar.HitArea Date;

[VB] Public Const Date As MonthCalendar.HitArea

[JScript] public var Date : MonthCalendar.HitArea;

*Description*

The specified point is on a date within the calendar. The  
**System.Windows.Forms.MonthCalendar.HitTestInfo.Time** property of  
**System.Windows.Forms.MonthCalendar.HitTestInfo** is set to the date at  
the specified point.

d) *ToString*

```
[C#]      public      const      MonthCalendar.HitArea      DayOfWeek;  
[C++]     public:     const      MonthCalendar.HitArea      DayOfWeek;  
[VB]      Public      Const      DayOfWeek      As      MonthCalendar.HitArea  
[JScript] public      var      DayOfWeek      :      MonthCalendar.HitArea;
```

*Description*

The specified point is over a day abbreviation ("Fri", for example). The **System.Windows.Forms.MonthCalendar.HitTestInfo.Time** property of **System.Windows.Forms.MonthCalendar.HitTestInfo** is set to the corresponding date on the top row.

e) *ToString*

```
[C#]      public      const      MonthCalendar.HitArea      NextMonthButton;  
[C++]     public:     const      MonthCalendar.HitArea      NextMonthButton;  
[VB]      Public      Const      NextMonthButton      As      MonthCalendar.HitArea  
[JScript] public      var      NextMonthButton      :      MonthCalendar.HitArea;
```

*Description*

The specified point is over the button at the upper-right corner of the control. If the user clicks here, the month calendar will scroll its display to the next month or set of months.

f) *ToString*

```
[C#]      public      const      MonthCalendar.HitArea      NextMonthDate;  
[C++]     public:     const      MonthCalendar.HitArea      NextMonthDate;
```

```

1 [VB]    Public    Const    NextMonthDate    As    MonthCalendar.HitArea
2 [JScript]    public    var    NextMonthDate    :    MonthCalendar.HitArea;

```

#### *Description*

The specified point is over a date from the next month (partially displayed at the end of the currently displayed month). If the user clicks here, the month calendar scrolls its display to the next month or set of months.

#### *g) ToString*

```

9 [C#]    public    const    MonthCalendar.HitArea    Nowhere;
10 [C++]    public:    const    MonthCalendar.HitArea    Nowhere;
11 [VB]    Public    Const    Nowhere    As    MonthCalendar.HitArea
12 [JScript]    public    var    Nowhere    :    MonthCalendar.HitArea;

```

#### *Description*

The specified point is not on the month calendar control, or it is in an inactive portion of the control.

#### *h) ToString*

```

19 [C#]    public    const    MonthCalendar.HitArea    PrevMonthButton;
20 [C++]    public:    const    MonthCalendar.HitArea    PrevMonthButton;
21 [VB]    Public    Const    PrevMonthButton    As    MonthCalendar.HitArea
22 [JScript]    public    var    PrevMonthButton    :    MonthCalendar.HitArea;

```

#### *Description*

The specified point is over the button at the upper-left corner of the control. If the user clicks here, the month calendar will scroll its display to the previous

month or set of months The given point was over the button at the top left corner of the control. If the user clicks here, the month calendar will scroll its display to the previous month or set of months

*i) ToString*

```
[C#]      public      const      MonthCalendar.HitArea      PrevMonthDate;
[C++]     public:     const      MonthCalendar.HitArea      PrevMonthDate;
[VB]      Public      Const      PrevMonthDate      As      MonthCalendar.HitArea
[JScript] public      var      PrevMonthDate      :      MonthCalendar.HitArea;
```

*Description*

The specified point is over a date from the previous month (partially displayed at the end of the currently displayed month). If the user clicks here, the month calendar scrolls its display to the previous month or set of months.

*j) ToString*

```
[C#]      public      const      MonthCalendar.HitArea      TitleBackground;
[C++]     public:     const      MonthCalendar.HitArea      TitleBackground;
[VB]      Public      Const      TitleBackground      As      MonthCalendar.HitArea
[JScript] public      var      TitleBackground      :      MonthCalendar.HitArea;
```

*Description*

The specified point is over the background of a month's title.

*k) ToString*

```
[C#]      public      const      MonthCalendar.HitArea      TitleMonth;
[C++]     public:     const      MonthCalendar.HitArea      TitleMonth;
```

```

1  [VB]      Public      Const      TitleMonth      As      MonthCalendar.HitArea
2  [JScript] public      var      TitleMonth      :      MonthCalendar.HitArea;

```

#### *Description*

The specified point is in a month's title bar, over a month name.

#### *l) ToString*

```

8  [C#]      public      const      MonthCalendar.HitArea      TitleYear;
9  [C++]     public:     const      MonthCalendar.HitArea      TitleYear;
10 [VB]      Public      Const      TitleYear      As      MonthCalendar.HitArea
11 [JScript] public      var      TitleYear      :      MonthCalendar.HitArea;

```

#### *Description*

The specified point is in a month's title bar, over the year value.

#### *m) ToString*

```

17 [C#]      public      const      MonthCalendar.HitArea      TodayLink;
18 [C++]     public:     const      MonthCalendar.HitArea      TodayLink;
19 [VB]      Public      Const      TodayLink      As      MonthCalendar.HitArea
20 [JScript] public      var      TodayLink      :      MonthCalendar.HitArea;

```

#### *Description*

The specified point is on the "today" link at the bottom of the month calendar control.

n) *ToString*

```
[C#]      public      const      MonthCalendar.HitArea      WeekNumbers;
[C++]     public:     const      MonthCalendar.HitArea      WeekNumbers;
[VB]      Public      Const      WeekNumbers      As      MonthCalendar.HitArea
[JScript] public      var      WeekNumbers      :      MonthCalendar.HitArea;
```

*Description*

The specified point is over a week number. This will only occur if the **System.Windows.Forms.MonthCalendar.ShowWeekNumbers** property of **System.Windows.Forms.MonthCalendar** is enabled. The **System.Windows.Forms.MonthCalendar.HitTestInfo.Time** property of **System.Windows.Forms.MonthCalendar.HitTestInfo** is set to the corresponding date in the leftmost column.

DataGrid.HitTestInfo class (System.Windows.Forms)

a) *ToString*

*Description*

Contains information about a part of the **System.Windows.Forms.DataGrid** at a specified coordinate. This class cannot be inherited.

The **System.Windows.Forms.DataGrid.HitTestInfo** class, in conjunction with the **System.Windows.Forms.DataGrid.HitTest(System.Int32, System.Int32)** method of the **System.Windows.Forms.DataGrid** control, is used to determine which part of a **System.Windows.Forms.DataGrid** control the user has clicked. The **System.Windows.Forms.DataGrid.HitTestInfo** contains both the row, column and part of the grid that was clicked. See the **System.Windows.Forms.DataGrid.HitTestType** enumeration returned by the **System.Windows.Forms.DataGrid.HitTestInfo.Type** property for a complete list of grid parts.

b) *ToString*

```
[C#]      public      static      readonly      DataGrid.HitTestInfo      Nowhere;  
[C++]      public:      static      DataGrid.HitTestInfo*      Nowhere;  
[VB]      Public      Shared      ReadOnly      Nowhere      As      DataGrid.HitTestInfo  
[JScript]      public      static      var      Nowhere      :      DataGrid.HitTestInfo;
```

*Description*

Indicates that a coordinate corresponds to a non-functioning part of the **System.Windows.Forms.DataGrid** control.

Other parts of the **System.Windows.Forms.DataGrid** , such as the **System.Windows.Forms.DataGrid.HitTestType.Caption** , can return useful information. If the part of the grid has no function (such as the gray area behind a sparsely populated grid table), the **System.Windows.Forms.DataGrid.HitTestInfo.Nowhere** field is returned.

c) *Column*

d) *ToString*

```
[C#]      public      int      Column      {get;}  
[C++]      public:      __property      int      get_Column();  
[VB]      Public      ReadOnly      Property      Column      As      Integer  
[JScript]      public      function      get      Column()      :      int;
```

*Description*

Gets the number of the column the user has clicked.

If the coordinate is not a cell, the property returns -1.

e) *Row*

f) *ToString*

```
[C#]          public          int          Row          {get;}
[C++]          public:          __property          int          get_Row();
[VB]          Public          ReadOnly          Property          Row          As          Integer
[JScript]          public          function          get          Row()          :          int;
```

*Description*

Gets the number of the row the user has clicked.

If the coordinate is not a cell, the property returns -1. If the coordinate is a **System.Windows.Forms.DataGrid.HitTestType.RowHeader**, the property returns the row number of the header, but the **System.Windows.Forms.DataGrid.HitTestInfo.Column** property will return -1.

g) *Type*

h) *ToString*

```
[C#]          public          DataGrid.HitTestType          Type          {get;}
[C++]          public:          __property          DataGrid.HitTestType          get_Type();
[VB]          Public          ReadOnly          Property          Type          As          DataGrid.HitTestType
[JScript]          public          function          get          Type()          :          DataGrid.HitTestType;
```

*Description*

Gets the part of the **System.Windows.Forms.DataGrid** control, other than the row or column, that was clicked.



i) *Equals*

[C#] public override bool Equals(object value);  
[C++] public: bool Equals(Object\* value);  
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean  
[JScript] public override function Equals(value : Object) : Boolean;

*Description*

Indicates whether two objects are identical.

*Return Value:* **true** if the objects are equal; otherwise, **false** . The second object to compare, typed as **System.Object**.

j) *GetHashCode*

[C#] public override int GetHashCode();  
[C++] public: int GetHashCode();  
[VB] Overrides Public Function GetHashCode() As Integer  
[JScript] public override function GetHashCode() : int;

*Description*

Gets the hash code for the **System.Windows.Forms.DataGrid.HitTestInfo** instance.

*Return Value:* The hash code for this instance.

This method overrides **System.Object.GetHashCode** .

k) *ToString*

[C#] public override string ToString();  
[C++] public: String\* ToString();

[VB] Overrides Public Function ToString() As String  
 [JScript] public override function ToString() : String;

#### Description

Gets the type, row number, and column number.

*Return Value:* The type, row number, and column number.

MonthCalendar.HitTestInfo class (System.Windows.Forms)

a) *ToString*

#### Description

**System.Windows.Forms.MonthCalendar.HitTestInfo** objects are returned by **System.Windows.Forms.MonthCalendar** in response to the **System.Windows.Forms.MonthCalendar.HitTest(System.Int32, System.Int32)** method.

b) *HitArea*

c) *ToString*

[C#] public MonthCalendar.HitArea HitArea {get;}

[C++] public: \_\_property MonthCalendar.HitArea get\_HitArea();

[VB] Public ReadOnly Property HitArea As MonthCalendar.HitArea

[JScript] public function get HitArea() : MonthCalendar.HitArea;

#### Description

Output member that receives an enumeration value from

**System.Windows.Forms.MonthCalendar.HitArea** representing the result of the hit-test operation.

d) *Point*

e) *ToString*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | Point      | Point    | {get;}           |
| [C++]     | public: | __property | Point    | get_Point();     |
| [VB]      | Public  | ReadOnly   | Property | Point As Point   |
| [JScript] | public  | function   | get      | Point() : Point; |

*Description*

The point that was hit-tested.

f) *Time*

g) *ToString*

|           |         |            |          |                    |
|-----------|---------|------------|----------|--------------------|
| [C#]      | public  | DateTime   | Time     | {get;}             |
| [C++]     | public: | __property | DateTime | get_Time();        |
| [VB]      | Public  | ReadOnly   | Property | Time As DateTime   |
| [JScript] | public  | function   | get      | Time() : DateTime; |

*Description*

The time information specific to the location that was hit-tested. This value will only be valid at certain values of

**System.Windows.Forms.MonthCalendar.HitTestInfo.HitArea .**

## DataGrid.HitTestType enumeration (System.Windows.Forms)

### a) ToString

#### Description

Specifies the part of the **System.Windows.Forms.DataGrid** control the user has clicked.

Use the members of this enumeration to determine which part of the grid has been clicked. The **System.Windows.Forms.DataGrid.HitTestInfo.Type** property of a **System.Windows.Forms.DataGrid.HitTestInfo** object returns a **System.Windows.Forms.DataGrid.HitTestType**. The **System.Windows.Forms.DataGrid.HitTestInfo** is created by invoking the **System.Windows.Forms.DataGrid.HitTest(System.Int32, System.Int32)** method of a **System.Windows.Forms.DataGrid** control.

### b) ToString

|           |         |       |                      |                         |
|-----------|---------|-------|----------------------|-------------------------|
| [C#]      | public  | const | DataGrid.HitTestType | Caption;                |
| [C++]     | public: | const | DataGrid.HitTestType | Caption;                |
| [VB]      | Public  | Const | Caption              | As DataGrid.HitTestType |
| [JScript] | public  | var   | Caption              | : DataGrid.HitTestType; |

#### Description

The caption of the **System.Windows.Forms.DataGrid** control.

### c) ToString

|       |         |       |                      |                         |
|-------|---------|-------|----------------------|-------------------------|
| [C#]  | public  | const | DataGrid.HitTestType | Cell;                   |
| [C++] | public: | const | DataGrid.HitTestType | Cell;                   |
| [VB]  | Public  | Const | Cell                 | As DataGrid.HitTestType |

1 [JScript] public var Cell : DataGrid.HitTestType;

3 *Description*

4 A cell in the **System.Windows.Forms.DataGrid** control.

5 *d) ToString*

7 [C#] public const DataGrid.HitTestType ColumnHeader;

8 [C++] public: const DataGrid.HitTestType ColumnHeader;

9 [VB] Public Const ColumnHeader As DataGrid.HitTestType

10 [JScript] public var ColumnHeader : DataGrid.HitTestType;

12 *Description*

13 A column header in the **System.Windows.Forms.DataGrid** control.

14 *e) ToString*

16 [C#] public const DataGrid.HitTestType ColumnResize;

17 [C++] public: const DataGrid.HitTestType ColumnResize;

18 [VB] Public Const ColumnResize As DataGrid.HitTestType

19 [JScript] public var ColumnResize : DataGrid.HitTestType;

21 *Description*

22 The column border, which is the line between column headers. It can be  
23 dragged to resize a column's width.  
24  
25

### f) ToString

|           |         |       |                      |                         |
|-----------|---------|-------|----------------------|-------------------------|
| [C#]      | public  | const | DataGrid.HitTestType | None;                   |
| [C++]     | public: | const | DataGrid.HitTestType | None;                   |
| [VB]      | Public  | Const | None                 | As DataGrid.HitTestType |
| [JScript] | public  | var   | None                 | : DataGrid.HitTestType; |

#### Description

The background area, visible when the control contains no table, few rows, or when a table is scrolled to its bottom.

### g) ToString

|           |         |       |                      |                         |
|-----------|---------|-------|----------------------|-------------------------|
| [C#]      | public  | const | DataGrid.HitTestType | ParentRows;             |
| [C++]     | public: | const | DataGrid.HitTestType | ParentRows;             |
| [VB]      | Public  | Const | ParentRows           | As DataGrid.HitTestType |
| [JScript] | public  | var   | ParentRows           | : DataGrid.HitTestType; |

#### Description

The parent row section of the **System.Windows.Forms.DataGrid** control. The parent row displays information from or about the parent table of the currently displayed child table, such as the name of the parent table, column names and values of the parent record.

### h) ToString

|       |         |       |                      |                         |
|-------|---------|-------|----------------------|-------------------------|
| [C#]  | public  | const | DataGrid.HitTestType | RowHeader;              |
| [C++] | public: | const | DataGrid.HitTestType | RowHeader;              |
| [VB]  | Public  | Const | RowHeader            | As DataGrid.HitTestType |

1 [JScript]     public     var     RowHeader     :     DataGrid.HitTestType;

2  
3 *Description*

4 A row header in the **System.Windows.Forms.DataGrid** control.

5         *i)     ToString*

6  
7 [C#]         public         const         DataGrid.HitTestType         RowResize;

8 [C++]         public:         const         DataGrid.HitTestType         RowResize;

9 [VB]         Public         Const         RowResize         As         DataGrid.HitTestType

10 [JScript]     public     var     RowResize     :     DataGrid.HitTestType;

11  
12 *Description*

13 The row border, which is the line between grid row headers. It can be dragged to resize a row's height.

14         HorizontalAlignment enumeration (System.Windows.Forms)

15  
16         *a)     ToString*

17  
18  
19 *Description*

20 Specifies how an object or text in a control is horizontally aligned relative to an element of the control.

21 This enumeration is used in numerous classes. A partial list of these classes is  
22 **System.Windows.Forms.CheckedListBox** ,  
23 **System.Windows.Forms.ColumnHeader** ,  
24 **System.Windows.Forms.ComboBox** ,  
25 **System.Windows.Forms.ControlPaint** , **System.Windows.Forms.Label** ,  
**System.Windows.Forms.ListBox** , **System.Windows.Forms.Control** ,  
**System.Windows.Forms.RichTextBox** , and  
**System.Windows.Forms.TextBox** .

**b) ToString**

|           |         |       |                     |    |                      |
|-----------|---------|-------|---------------------|----|----------------------|
| [C#]      | public  | const | HorizontalAlignment |    | Center;              |
| [C++]     | public: | const | HorizontalAlignment |    | Center;              |
| [VB]      | Public  | Const | Center              | As | HorizontalAlignment  |
| [JScript] | public  | var   | Center              | :  | HorizontalAlignment; |

*Description*

The object or text is aligned in the center of the control element.

**c) ToString**

|           |         |       |                     |    |                      |
|-----------|---------|-------|---------------------|----|----------------------|
| [C#]      | public  | const | HorizontalAlignment |    | Left;                |
| [C++]     | public: | const | HorizontalAlignment |    | Left;                |
| [VB]      | Public  | Const | Left                | As | HorizontalAlignment  |
| [JScript] | public  | var   | Left                | :  | HorizontalAlignment; |

*Description*

The object or text is aligned on the left of the control element.

**d) ToString**

|           |         |       |                     |    |                      |
|-----------|---------|-------|---------------------|----|----------------------|
| [C#]      | public  | const | HorizontalAlignment |    | Right;               |
| [C++]     | public: | const | HorizontalAlignment |    | Right;               |
| [VB]      | Public  | Const | Right               | As | HorizontalAlignment  |
| [JScript] | public  | var   | Right               | :  | HorizontalAlignment; |



1  
2 *Description*

3 The object or text is aligned on the right of the control element.

4 HScrollBar class (System.Windows.Forms)

5 *a) ToString*

6  
7  
8 *Description*

9 Represents a standard Windows horizontal scroll bar.

10 Most controls that need scroll bars already provide them and do not require this  
11 control. This is true of a multi-line **System.Windows.Forms.TextBox** control,  
12 a **System.Windows.Forms.ListBox** , and a  
13 **System.Windows.Forms.ComboBox** , for example.

14 *b) HScrollBar*

15 *Example Syntax:*

16 *c) ToString*

17 [C#] public HScrollBar();  
18 [C++] public: HScrollBar();  
19 [VB] Public Sub New()  
20 [JScript] public function HScrollBar();  
21  
22  
23  
24  
25

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *Bottom*
- o) *Bounds*
- p) *CanFocus*
- q) *CanSelect*
- r) *Capture*
- s) *CausesValidation*
- t) *ClientRectangle*
- u) *ClientSize*
- v) *CompanyName*
- w) *Container*
- x) *ContainsFocus*
- y) *ContextMenu*
- z) *Controls*

1        **aa)    *Created***

2        **bb)    *CreateParams***

3        **cc)    *ToString***

4  
5  
6        ***Description***

7        Returns the parameters needed to create the handle. Inheriting classes can  
8        override this to provide extra functionality. They should not, however, forget to  
9        call `base.getCreateParams()` first to get the struct filled up with the basic info.

9        **dd)    *Cursor***

10       **ee)    *DataBindings***

11       **ff)    *DefaultImeMode***

12       **gg)    *DefaultSize***

13       **hh)    *ToString***

14  
15  
16       ***Description***

17       Deriving classes can override this to configure a default size for their control.  
18       This is more efficient than setting the size in the control's constructor.

- 1      **ii)      *DesignMode***
- 2      **jj)      *DisplayRectangle***
- 3      **kk)      *Disposing***
- 4      **ll)      *Dock***
- 5      **mm)      *Enabled***
- 6      **nn)      *Events***
- 7      **oo)      *Focused***
- 8      **pp)      *Font***
- 9      **qq)      *FontHeight***
- 10      **rr)      *ForeColor***
- 11      **ss)      *Handle***
- 12      **tt)      *HasChildren***
- 13      **uu)      *Height***
- 14      **vv)      *ImeMode***
- 15      **ww)      *InvokeRequired***
- 16      **xx)      *IsAccessible***
- 17      **yy)      *IsDisposed***
- 18      **zz)      *IsHandleCreated***
- 19      **aaa)      *LargeChange***
- 20      **bbb)      *Left***
- 21      **ccc)      *Location***
- 22      **ddd)      *Maximum***
- 23      **eee)      *Minimum***
- 24
- 25

1 *fff) Name*  
 2 *ggg) Parent*  
 3 *hhh) ProductName*  
 4 *iii) ProductVersion*  
 5 *jjj) RecreatingHandle*  
 6 *kkk) Region*  
 7 *lll) RenderRightToLeft*  
 8 *mmm) ResizeRedraw*  
 9 *nnn) Right*  
 10 *ooo) RightToLeft*  
 11 *ppp) ShowFocusCues*  
 12 *qqq) ShowKeyboardCues*  
 13 *rrr) Site*  
 14 *sss) Size*  
 15 *ttt) SmallChange*  
 16 *uuu) TabIndex*  
 17 *vvv) TabStop*  
 18 *www) Tag*  
 19 *xxx) Text*  
 20 *yyy) Top*  
 21 *zzz) TopLevelControl*  
 22 *aaaa) Value*  
 23 *bbbb) Visible*

cccc) *Width*

dddd) *WindowTarget*

IButtonControl interface (System.Windows.Forms)

a) *WndProc*

### *Description*

Allows a control to act like a button on a form.

An example of where this interface might be implemented is default and cancel button processing. Default buttons are notified when an unprocessed ENTER key is entered for a form, just like a dialog would be closed. Similarly, cancel buttons are notified whenever an unprocessed ESC key is entered on a form, much like a dialog would be dismissed.

b) *DialogResult*

c) *WndProc*

[C#]            DialogResult            DialogResult            {get;            set;}

[C++] DialogResult get\_DialogResult();void set\_DialogResult(DialogResult);

[VB]            Property            DialogResult            As            DialogResult

[JScript] abstract function get DialogResult() : DialogResult;public abstract

function                            set                            DialogResult(DialogResult);

### *Description*

Gets or sets the value returned to the parent form when the button is clicked.

When a form is shown as a dialog box using the **System.Windows.Forms.Form.ShowDialog** method and one of its buttons is clicked, the button's **System.Windows.Forms.IButtonControl.DialogResult** value is assigned to the form's **System.Windows.Forms.Form.DialogResult** property.

#### d) *NotifyDefault*

|           |          |                     |                   |
|-----------|----------|---------------------|-------------------|
| [C#]      | void     | NotifyDefault(bool  | value);           |
| [C++]     | void     | NotifyDefault(bool  | value);           |
| [VB]      | Sub      | NotifyDefault(ByVal | value As Boolean) |
| [JScript] | function | NotifyDefault(value | : Boolean);       |

#### *Description*

Notifies a control that it is the default button so that its appearance and behavior is adjusted accordingly.

This method is called by a parent form to make a control the default button. Default buttons are set to have an extra thick border. **true** if the control should behave as a default button; otherwise **false**.

#### e) *PerformClick*

|           |          |                 |
|-----------|----------|-----------------|
| [C#]      | void     | PerformClick(); |
| [C++]     | void     | PerformClick(); |
| [VB]      | Sub      | PerformClick()  |
| [JScript] | function | PerformClick(); |

#### *Description*

Generates a **System.Windows.Forms.Control.Click** event for the control.

This method is called for the button that has focus, or for the default button(if no other button has focus) when the user presses the ENTER key. This method is also called when the user presses the ESC key if the button is set as the cancel button.

1 ICommandExecutor interface (System.Windows.Forms)

2 **a) PerformClick**

5 *Description*

6 **b) Execute**

|              |          |            |
|--------------|----------|------------|
| 8 [C#]       | void     | Execute(); |
| 9 [C++]      | void     | Execute(); |
| 10 [VB]      | Sub      | Execute()  |
| 11 [JScript] | function | Execute(); |

13 *Description*

15 IComponentEditorPageSite interface (System.Windows.Forms)

16 **a) Execute**

19 *Description*

20 The site for a ComponentEditorPage.

21 **b) GetControl**

|          |          |               |
|----------|----------|---------------|
| 23 [C#]  | Control  | GetControl(); |
| 24 [C++] | Control* | GetControl(); |



|           |          |              |    |          |
|-----------|----------|--------------|----|----------|
| [VB]      | Function | GetControl() | As | Control  |
| [JScript] | function | GetControl() | :  | Control; |

#### Description

Returns the parent control for the page window.

#### c) *SetDirty*

|           |          |             |
|-----------|----------|-------------|
| [C#]      | void     | SetDirty(); |
| [C++]     | void     | SetDirty(); |
| [VB]      | Sub      | SetDirty()  |
| [JScript] | function | SetDirty(); |

#### Description

Notifies the site that the editor is in dirty state.

IContainerControl interface (System.Windows.Forms)

#### a) *SetDirty*

#### Description

Provides the functionality for a control to act as a parent for other controls.

Implement this interface in classes that you wish to parent a collection of controls. The members of this interface allow you to activate a child control, or determine which control is currently active. When implemented in a class, **System.Windows.Forms.IContainerControl.ActivateControl(System.Windows.Forms.Control)** takes a **System.Windows.Forms.Control** as a parameter and activates the specified control. The **System.Windows.Forms.IContainerControl.ActiveControl** property activates or retrieves the control that is active.

b) *ActiveControl*

c) *SetDirty*

```
[C#]          Control          ActiveControl          {get;          set;}
[C++]   Control*   get_ActiveControl();void   set_ActiveControl(Control*);
[VB]          Property          ActiveControl          As          Control
[JScript] abstract function get ActiveControl() : Control;public abstract function
set                                     ActiveControl(Control);
```

*Description*

Gets or sets the control that is active on the container control.

When implemented in a class, this property activates or retrieves the active control on the container control.

d) *ActivateControl*

```
[C#]          bool          ActivateControl(Control          active);
[C++]          bool          ActivateControl(Control*          active);
[VB] Function ActivateControl(ByVal active As Control) As Boolean
[JScript] function ActivateControl(active : Control) : Boolean;
```

*Description*

Activates a specified control.

*Return Value:* **true** if the control is successfully activated; otherwise, **false** .

When implemented in a class, this method activates the specified **System.Windows.Forms.Control** . The control must be a child of the container control. The **System.Windows.Forms.Control** being activated.

1           IDataGridColumnStyleEditingNotificationService           interface  
2 (System.Windows.Forms)

3           **a)     ActivateControl**

6    *Description*

7    Provides an editing notification interface.

8    To create a new **System.Windows.Forms.DataGrid** column with special  
9    properties, you must first inherit from the  
10   **System.Windows.Forms.DataGridColumnStyle** class. The  
11   **System.Windows.Forms.IDataGridColumnStyleEditingNotificationService**  
12   interface is provided to notify the **System.Windows.Forms.DataGrid** when  
13   a column is being edited.

12          **b)     ColumnStartedEditing**

14    [C#]       void       ColumnStartedEditing(Control       editingControl);  
15    [C++]       void       ColumnStartedEditing(Control\*     editingControl);  
16    [VB]    Sub   ColumnStartedEditing(ByVal   editingControl   As   Control)  
17    [JScript]   function   ColumnStartedEditing(editingControl   :   Control);

19    *Description*

20    Informs the **System.Windows.Forms.DataGrid** that the user has begun  
21    editing the column.

22    When called, the  
23    **System.Windows.Forms.IDataGridColumnStyleEditingNotificationService.ColumnStartedEditing(System.Windows.Forms.Control)** method  
24    allows the **System.Windows.Forms.DataGrid** control to show a pencil in the  
25    row header indicating the row is being edited. The  
26    **System.Windows.Forms.Control** that is editing the column.

IDataGridEditingService interface (System.Windows.Forms)

**a) ColumnStartedEditing**

*Description*

The DataGrid exposes hooks to request editing commands via this interface.

**b) BeginEdit**

[C#] bool BeginEdit(DataGridColumnStyle gridColumn, int rowNumber);

[C++] bool BeginEdit(DataGridColumnStyle\* gridColumn, int rowNumber);

[VB] Function BeginEdit(ByVal gridColumn As DataGridColumnStyle, ByVal  
rowNumber As Integer) As Boolean

[JScript] function BeginEdit(gridColumn : DataGridColumnStyle, rowNumber :  
int) : Boolean;

*Description*

Requests an edit operation.

*Return Value:* If the operation will be performed, true is returned; otherwise false is returned.

*Return Value:* If the operation will be performed, true is returned; otherwise false is returned. The DataGridColumn to edit. The number of the row to edit

**c) EndEdit**

[C#] bool EndEdit(DataGridColumnStyle gridColumn, int rowNumber, bool  
shouldAbort);

[C++] bool EndEdit(DataGridColumnStyle\* gridColumn, int rowNumber, bool

```

1  shouldAbort);
2  [VB] Function EndEdit(ByVal gridColumn As DataGridViewCellStyle, ByVal
3  rowNum As Integer, ByVal shouldAbort As Boolean) As Boolean
4  [JScript] function EndEdit(gridColumn : DataGridViewCellStyle, rowNum : int,
5  shouldAbort          :          Boolean)          :          Boolean;
6

```

### *Description*

Requests an end to an edit operation taking place.

*Return Value:* If the operation will be performed, true is returned; otherwise false is returned.

*Return Value:* If the operation will be performed, true is returned; otherwise false is returned. The DataGridViewColumn of the cell to cease editing. The number of the row to edit True if an abort operation is requested

IDataObject interface (System.Windows.Forms)

#### *a) EndEdit*

### *Description*

Provides a format-independent mechanism for transferring data.

The **IDataObject** interface is used by the **System.Windows.Forms.Clipboard** class and in drag-and-drop operations.

#### *b) GetData*

```

21  [C#]          object          GetData(string          format);
22  [C++]          Object*          GetData(String*          format);
23  [VB] Function  GetData(ByVal  format  As  String)  As  Object
24  [JScript] function  GetData(format  :  String)  :  Object;
25

```

## Description

Retrieves the data associated with the specified data format.

*Return Value:* The data associated with the specified format, or **null**.

If this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if the data was stored with `autoConvert` set to **false**, this method returns **null**.

The format of the data to retrieve. See

**System.Windows.Forms.DataFormats** for predefined formats.

### c) *GetData*

|           |          |                               |           |
|-----------|----------|-------------------------------|-----------|
| [C#]      | object   | GetData(Type                  | format);  |
| [C++]     | Object*  | GetData(Type*                 | format);  |
| [VB]      | Function | GetData(ByVal format As Type) | As Object |
| [JScript] | function | GetData(format : Type)        | : Object; |

## Description

Retrieves the data associated with the specified class type format.

*Return Value:* The data associated with the specified format, or **null**.

If this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if the data was stored with `autoConvert` set to **false**, this method returns **null**. A

**System.Type** representing the format of the data to retrieve. See

**System.Windows.Forms.DataFormats** for predefined formats.

### d) *GetData*

|       |          |                                                               |                    |
|-------|----------|---------------------------------------------------------------|--------------------|
| [C#]  | object   | GetData(string format,                                        | bool autoConvert); |
| [C++] | Object*  | GetData(String* format,                                       | bool autoConvert); |
| [VB]  | Function | GetData(ByVal format As String, ByVal autoConvert As Boolean) |                    |

As Object

[JScript] function GetData(format : String, autoConvert : Boolean) : Object;

Retrieves the data associated with the specified data format.

#### *Description*

Retrieves the data associated with the specified data format, using a Boolean to determine whether to convert the data to the format.

*Return Value:* The data associated with the specified format, or **null** .

If the *autoConvert* parameter is **true** and this method cannot find data in the specified format, it attempts to convert the data to the format. If the data cannot be converted to the specified format, or if the data was stored with the *autoConvert* parameter set to **false** , this method returns **null** . The format of the data to retrieve. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to convert the data to the specified format; otherwise, **false**.

#### *e)      GetDataPresent*

[C#]                      bool                      GetDataPresent(string                      format);

[C++]                      bool                      GetDataPresent(String\*                      format);

[VB]    Function    GetDataPresent(ByVal    format    As    String)    As    Boolean

[JScript]    function    GetDataPresent(format    :    String)    :    Boolean;

#### *Description*

Determines whether data stored in this instance is associated with, or can be converted to, the specified format.

*Return Value:* **true** if data stored in this instance is associated with, or can be converted to, the specified format; otherwise **false** .

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.IDataObject.GetData(System.String,System.Boolean)** . Call **System.Windows.Forms.IDataObject.GetFormats(System.Boolean)** for

the formats that are available in this instance. The format for which to check.  
See **System.Windows.Forms.DataFormats** for predefined formats.

*f) GetDataPresent*

```
[C#]          bool          GetDataPresent(Type          format);
[C++]          bool          GetDataPresent(Type*          format);
[VB] Function  GetDataPresent(ByVal format As Type) As Boolean
[JScript] function  GetDataPresent(format : Type) : Boolean;
```

*Description*

Determines whether data stored in this instance is associated with, or can be converted to, the specified format.

**Return Value:** **true** if data stored in this instance is associated with, or can be converted to, the specified format; otherwise, **false**.

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.IDataObject.GetData(System.String,System.Boolean)**. Call **System.Windows.Forms.IDataObject.GetFormats(System.Boolean)** for the formats that are available in this instance. A **System.Type** representing the format for which to check. See **System.Windows.Forms.DataFormats** for predefined formats.

*g) GetDataPresent*

```
[C#]    bool    GetDataPresent(string    format,    bool    autoConvert);
[C++]    bool    GetDataPresent(String*    format,    bool    autoConvert);
[VB] Function  GetDataPresent(ByVal format As String, ByVal autoConvert As
Boolean)          As          Boolean
[JScript] function  GetDataPresent(format : String, autoConvert : Boolean) :
Boolean; Determines whether data stored in this instance is associated with the
```



specified format.

### Description

Determines whether data stored in this instance is associated with the specified format, using a Boolean value to determine whether to convert the data to the format.

*Return Value:* **true** if the data is in, or can be converted to, the specified format; otherwise, **false**.

Call this method to determine whether a format exists in this **System.Windows.Forms.DataObject** instance before calling **System.Windows.Forms.IDataObject.GetData(System.String,System.Boolean)**. Call **System.Windows.Forms.IDataObject.GetFormats(System.Boolean)** for the formats that are available in this instance. The format for which to check. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to determine whether data stored in this instance can be converted to the specified format; **false** to check whether the data is in the specified format.

### h) GetFormats

|           |          |              |              |               |
|-----------|----------|--------------|--------------|---------------|
| [C#]      |          | string[]     |              | GetFormats(); |
| [C++]     | String*  |              | GetFormats() | __gc[];       |
| [VB]      | Function | GetFormats() | As           | String()      |
| [JScript] | function | GetFormats() | :            | String[];     |

### Description

Gets a list of all formats that data stored in this instance is associated with or can be converted to.

*Return Value:* An array of the names that represents a list of all formats that are supported by the data stored in this object.

Call this method to get the supported data formats before calling **System.Windows.Forms.IDataObject.GetData(System.String,System.Boolean)**. See **System.Windows.Forms.DataFormats** for the predefined formats.

### i) *GetFormats*

[C#]                    string[]                    GetFormats(bool                    autoConvert);  
[C++]                String\*                    GetFormats(bool                    autoConvert)                    \_\_gc[];  
[VB]    Function    GetFormats(ByVal    autoConvert    As    Boolean)    As    String()  
[JScript]    function    GetFormats(autoConvert : Boolean) : String[]; Gets a list of all  
formats that data stored in this instance is associated with or can be converted to.

#### *Description*

Gets a list of all formats that data stored in this instance is associated with or can be converted to, using a Boolean value to determine whether to retrieve all formats that the data can be converted to or only native data formats.

*Return Value:* An array of the names that represents a list of all formats that are supported by the data stored in this object.

Call this method to get the supported data formats before calling **System.Windows.Forms.IDataObject.GetData(System.String, System.Boolean)**. See **System.Windows.Forms.DataFormats** for the predefined formats. **true** to retrieve all formats that data stored in this instance is associated with, or can be converted to; **false** to retrieve only native data formats.

### j) *SetData*

[C#]                    void                    SetData(object                    data);  
[C++]                    void                    SetData(Object\*                    data);  
[VB]                Sub                    SetData(ByVal                    data                    As                    Object)  
[JScript]                function                    SetData(data                    :                    Object);

#### *Description*

Stores the specified data in this instance, using the class of the data for the format.

The data format is its class. If you do not know the format of the target application, you can store data in multiple formats using this method. The data to store.

*k) SetData*

```
[C#]      void      SetData(string      format,      object      data);  
[C++]     void      SetData(String*     format,      Object*     data);  
[VB]      Sub      SetData(ByVal format As String, ByVal data As Object)  
[JScript] function SetData(format : String, data : Object);
```

*Description*

Stores the specified data and its associated format in this instance.

If you do not know the format of the target application, you can store data in multiple formats using this method. The format associated with the data. See **System.Windows.Forms.DataFormats** for predefined formats. The data to store.

*l) SetData*

```
[C#]      void      SetData(Type      format,      object      data);  
[C++]     void      SetData(Type*     format,      Object*     data);  
[VB]      Sub      SetData(ByVal format As Type, ByVal data As Object)  
[JScript] function SetData(format : Type, data : Object);
```

*Description*

Stores the specified data and its associated class type in this instance.

If you do not know the format of the target application, you can store data in multiple formats using this method. A **System.Type** representing the format associated with the data. See **System.Windows.Forms.DataFormats** for predefined formats. The data to store.

### *m) SetData*

[C#] void SetData(string format, bool autoConvert, object data);

[C++] void SetData(String\* format, bool autoConvert, Object\* data);

[VB] Sub SetData(ByVal format As String, ByVal autoConvert As Boolean,  
ByVal data As Object)

[JScript] function SetData(format : String, autoConvert : Boolean, data : Object);

Stores the specified data and its associated format in this instance.

#### *Description*

Stores the specified data and its associated format in this instance, using a Boolean value to specify whether the data can be converted to another format.

If you do not know the format of the target application, you can store data in multiple formats using this method. The format associated with the data. See **System.Windows.Forms.DataFormats** for predefined formats. **true** to allow the data to be converted to another format; otherwise, **false**. The data to store.

IFeatureSupport interface (System.Windows.Forms)

### *a) SetData*

#### *Description*

Specifies a standard interface for retrieving feature information from the current system.

When implemented in a class, **System.Windows.Forms.IFeatureSupport** provides methods you can use to determine whether a feature is currently installed on the system and to get the version number of a feature. Call **System.Windows.Forms.IFeatureSupport.IsPresent(System.Object)** to determine whether a feature, or a specific version of a feature, is currently installed. Call **System.Windows.Forms.IFeatureSupport.GetVersionPresent(System.Object)** to determine the version number of an installed feature.

b) *GetVersionPresent*

```
[C#]          Version          GetVersionPresent(object          feature);
[C++]          Version*          GetVersionPresent(Object*          feature);
[VB] Function GetVersionPresent(ByVal feature As Object) As Version
[JScript] function GetVersionPresent(feature : Object) : Version;
```

*Description*

Retrieves the version of the specified feature.

*Return Value:* A **System.Version** representing the version number of the specified feature; or **null** if the feature is not installed.

For an implementation of this method, see

**System.Windows.Forms.OSFeature.GetVersionPresent(System.Object)**  
. The feature whose version is requested.

c) *IsPresent*

```
[C#]          bool          IsPresent(object          feature);
[C++]          bool          IsPresent(Object*          feature);
[VB] Function IsPresent(ByVal feature As Object) As Boolean
[JScript] function IsPresent(feature : Object) : Boolean; Determines whether the
specified feature is currently available on the system.
```

*Description*

Determines whether any version of the specified feature is currently available on the system.

*Return Value:* **true** if the feature is present; otherwise, **false** . The feature to look for.

d) *IsPresent*

```
[C#]    bool    IsPresent(object    feature,    Version    minimumVersion);  
[C++]   bool    IsPresent(Object*    feature,    Version*    minimumVersion);  
[VB] Function IsPresent(ByVal feature As Object, ByVal minimumVersion As  
Version)                                     As                                     Boolean  
[JScript] function IsPresent(feature : Object, minimumVersion : Version) :  
Boolean;
```

*Description*

Determines whether the specified or newer version of the specified feature is currently available on the system.

*Return Value:* **true** if the requested version of the feature is present; otherwise, **false**. The feature to look for. A **System.Version** representing the minimum version number of the feature to look for.

IFileReaderService interface (System.Windows.Forms)

a) *IsPresent*

*Description*

b) *OpenFileFromSource*

```
[C#]      Stream      OpenFileFromSource(string      relativePath);  
[C++]     Stream*     OpenFileFromSource(String*     relativePath);  
[VB] Function OpenFileFromSource(ByVal relativePath As String) As Stream  
[JScript] function OpenFileFromSource(relativePath : String) : Stream;
```

*Description*

ImageList.ImageCollection class (System.Windows.Forms)

*a) OpenFileFromSource*

*Description*

Encapsulates the collection of **System.Drawing.Image** objects in an **System.Windows.Forms.ImageList**.

This is used to manage the images in the **System.Windows.Forms.ImageList** programmatically, providing methods to add and remove image objects.

*b) Count*

*c) OpenFileFromSource*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | int        | Count    | {get;}           |
| [C++]     | public: | __property | int      | get_Count();     |
| [VB]      | Public  | ReadOnly   | Property | Count As Integer |
| [JScript] | public  | function   | get      | Count() : int;   |

*Description*

Gets the number of images currently in the list.

Counting the images forces the **System.Windows.Forms.ImageList.Handle** to be created.

d) *Empty*

e) *OpenFileFromSource*

|           |         |            |          |                    |
|-----------|---------|------------|----------|--------------------|
| [C#]      | public  | bool       | Empty    | {get;}             |
| [C++]     | public: | __property | bool     | get_Empty();       |
| [VB]      | Public  | ReadOnly   | Property | Empty As Boolean   |
| [JScript] | public  | function   | get      | Empty() : Boolean; |

*Description*

Gets a value indicating whether the **System.Windows.Forms.ImageList** has any images.

You can check this property without forcing the creation of a **System.Windows.Forms.ImageList.Handle**.

f) *IsReadOnly*

g) *OpenFileFromSource*

|           |         |            |            |                         |
|-----------|---------|------------|------------|-------------------------|
| [C#]      | public  | bool       | IsReadOnly | {get;}                  |
| [C++]     | public: | __property | bool       | get_IsReadOnly();       |
| [VB]      | Public  | ReadOnly   | Property   | IsReadOnly As Boolean   |
| [JScript] | public  | function   | get        | IsReadOnly() : Boolean; |

*Description*

Gets a value indicating whether the list is read only.



h) *Item*

i) *OpenFileFromSource*

```
[C#]      public      Image      this[int      index]      {get;      set;}
[C++] public: __property Image* get_Item(int index);public: __property void
set_Item(int              index,              Image*);
[VB] Public Default Property Item(ByVal index As Integer) As Image
[JScript]              returnValue              =
ImageCollectionObject.Item(index);ImageCollectionObject.Item(index)      =
returnValue;
```

*Description*

Gets or sets an **System.Drawing.Image** in an existing **System.Windows.Forms.ImageList** . The index of the image to get or set.

j) *Add*

```
[C#]      public      void      Add(Icon      value);
[C++]      public:      void      Add(Icon*      value);
[VB]      Public      Sub      Add(ByVal      value      As      Icon)
[JScript] public function Add(value : Icon); Adds the specified object to the
System.Windows.Forms.ImageList .
```

*Description*

Adds the specified icon to the **System.Windows.Forms.ImageList** .

The **System.Drawing.Icon** is converted to a **System.Drawing.Bitmap** before it is added to the list. An **System.Drawing.Icon** to add to the list.

k) *Add*

```
1
2 [C#]          public          void          Add(Image          value);
3
4 [C++]          public:          void          Add(Image*          value);
5
6 [VB]          Public          Sub          Add(ByVal          value          As          Image)
7
8 [JScript]          public          function          Add(value          :          Image);
9
```

*Description*

Adds the specified image to the **System.Windows.Forms.ImageList** . A **System.Drawing.Bitmap** of the image to add to the list.

l) *Add*

```
10
11
12 [C#]          public          int          Add(Image          value,          Color          transparentColor);
13
14 [C++]          public:          int          Add(Image*          value,          Color          transparentColor);
15
16 [VB]          Public          Function          Add(ByVal          value          As          Image,          ByVal          transparentColor          As
17          Color)
18          As          Integer
19
20 [JScript]          public          function          Add(value          :          Image,          transparentColor          :          Color)          :          int;
21
```

*Description*

Adds the specified image to the **System.Windows.Forms.ImageList** , using the specified color to generate the mask.

*Return Value:* The index of the newly added image or -1 if the the image could not be added. A **System.Drawing.Bitmap** of the image to add to the list. The **System.Drawing.Color** to mask this image.

m) *AddStrip*

```
22
23
24 [C#]          public          int          AddStrip(Image          value);
25
```

```

1 [C++]      public:      int      AddStrip(Image*      value);
2 [VB] Public Function AddStrip(ByVal value As Image) As Integer
3 [JScript] public function AddStrip(value : Image) : int;

```

#### *Description*

Adds an image strip to the specified image to the

**System.Windows.Forms.ImageList** .

*Return Value:* The index of the newly added image or -1 if the the image could not be added.

The number of images to add is inferred from the width of the given image. A **System.Drawing.Bitmap** object with the image(s) to add.

#### *n) Clear*

```

12 [C#]      public      void      Clear();
13 [C++]      public:      __sealed      void      Clear();
14 [VB]      NotOverridable      Public      Sub      Clear()
15 [JScript]      public      function      Clear();

```

#### *Description*

Removes all the images and masks from the

**System.Windows.Forms.ImageList** .

#### *o) Contains*

```

22 [C#]      public      bool      Contains(Image      image);
23 [C++]      public:      bool      Contains(Image*      image);
24 [VB] Public Function Contains(ByVal image As Image) As Boolean
25 [JScript] public function Contains(image : Image) : Boolean;

```

## Description

Not supported. The **System.Collections.IList.Contains** method indicates whether a specified object is contained in the list.

*Return Value:* **true** if the image is found in the list; otherwise, **false**.

This implementation of **System.Collections.IList.Contains** throws a **System.NotSupportedException** exception. The **System.Drawing.Image** to find in the list.

### p) GetEnumerator

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]         public:         __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JavaScript]   public         function          GetEnumerator()       :   IEnumerator;
```

## Description

Returns an enumerator that can be used to iterate through the item collection.

*Return Value:* An **System.Collections.IEnumerator** object that represents the item collection.

### q) IndexOf

```
[C#]          public          int          IndexOf(Image          image);
[C++]         public:         int          IndexOf(Image*          image);
[VB] Public Function IndexOf(ByVal image As Image) As Integer
[JavaScript]   public         function          IndexOf(image       :   Image)       :   int;
```

## Description

Not supported. The **System.Collections.IList.IndexOf** method returns the index of a specified object in the list.

*Return Value:* The index of the image in the list.

This implementation of **System.Collections.IList.IndexOf** throws a **System.NotSupportedException** exception. The **System.Drawing.Image** to find in the list.

*r) Remove*

[C#]            public            void            Remove(Image            image);

[C++]           public:            void            Remove(Image\*           image);

[VB]        Public        Sub        Remove(ByVal        image        As        Image)

[JScript]        public        function        Remove(image        :        Image);

*Description*

Not supported. The **System.Collections.IList.Remove** method removes a specified object from the list.

This implementation of **System.Collections.IList.Remove** throws a **System.NotSupportedException** exception. The **System.Drawing.Image** to remove from the list.

*s) RemoveAt*

[C#]            public            void            RemoveAt(int            index);

[C++]           public:            \_\_sealed        void            RemoveAt(int            index);

[VB]    NotOverridable    Public    Sub    RemoveAt(ByVal    index    As    Integer)

[JScript]        public        function        RemoveAt(index        :        int);

*Description*

Removes an image from the list. The index of the image to remove.

t) ***ICollection.CopyTo***

[C#] void ICollection.CopyTo(Array dest, int index);

[C++] void ICollection::CopyTo(Array\* dest, int index);

[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements  
ICollection.CopyTo

[JScript] function ICollection.CopyTo(dest : Array, index : int);

u) ***IList.Add***

[C#] int IList.Add(object value);

[C++] int IList::Add(Object\* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

v) ***IList.Contains***

[C#] bool IList.Contains(object image);

[C++] bool IList::Contains(Object\* image);

[VB] Function Contains(ByVal image As Object) As Boolean Implements  
IList.Contains

[JScript] function IList.Contains(image : Object) : Boolean;

w) ***IList.IndexOf***

[C#] int IList.IndexOf(object image);

[C++] int IList::IndexOf(Object\* image);

[VB] Function IndexOf(ByVal image As Object) As Integer Implements

1 IList.IndexOf

2 [JScript] function IList.IndexOf(image : Object) : int;

3 x) *IList.Insert*

5 [C#] void IList.Insert(int index, object value);

6 [C++] void IList::Insert(int index, Object\* value);

7 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

8 IList.Insert

9 [JScript] function IList.Insert(index : int, value : Object);

10 y) *IList.Remove*

12 [C#] void IList.Remove(object image);

13 [C++] void IList::Remove(Object\* image);

14 [VB] Sub Remove(ByVal image As Object) Implements IList.Remove

15 [JScript] function IList.Remove(image : Object);

16 ImageIndexConverter class (System.Windows.Forms)

17 a) *ToString*

20 *Description*

21 Provides a type converter to convert data for an image index to and from a  
22 string.

23 For more information about type converters, see the  
24 **System.ComponentModel.TypeConverter** base class and .

**b) *ImageIndexConverter***

*Example Syntax:*

**c) *ToString***

```
[C#] public ImageIndexConverter();
[C++] public: ImageIndexConverter();
[VB] Public Sub New()
[JavaScript] public function ImageIndexConverter();
```

**d) *IncludeNoneAsStandardValue***

**e) *ToString***

```
[C#] protected virtual bool IncludeNoneAsStandardValue {get;}
[C++] protected: __property virtual bool get_IncludeNoneAsStandardValue();
[VB] Overridable Protected ReadOnly Property IncludeNoneAsStandardValue As
Boolean
[JavaScript] protected function get IncludeNoneAsStandardValue() : Boolean;
```

***Description***

Gets or sets a value indicating whether a **none** or **null** value is valid in the **System.ComponentModel.TypeConverter.StandardValuesCollection** collection.

As implemented in this class is always returns **true** .

**f) *ConvertFrom***

```
[C#] public override object ConvertFrom(ITypeDescriptorContext context,
CultureInfo culture, object value);
```



```

1 [C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
2 CultureInfo* culture, Object* value);
3 [VB] Overrides Public Function ConvertFrom(ByVal context As
4 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
5 As Object
6 [JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
7 culture : CultureInfo, value : Object) : Object; Converts the given value to the
8 native type of the converter.

```

### *Description*

Converts the specified value object to a 32-bit signed integer object.

*Return Value:* An **System.Object** that represents the converted *value*.

This converter only can convert a 32-bit signed integer object to and from a string. An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Globalization.CultureInfo** object to provide locale information. The **System.Object** to convert.

### *g) ConvertTo*

```

17 [C#] public override object ConvertTo(ITypeDescriptorContext context,
18 CultureInfo culture, object value, Type destinationType);
19 [C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
20 culture, Object* value, Type* destinationType);
21 [VB] Overrides Public Function ConvertTo(ByVal context As
22 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
23 ByVal destinationType As Type) As Object
24 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,

```

1 culture : CultureInfo, value : Object, destinationType : Type) : Object; Converts  
2 the given value object to the specified destination type.

#### 4 *Description*

5 Converts the specified object to the destination type.

*Return Value:* An **System.Object** that represents the converted *value*.

6 Override this method to provide your own conversion requirements. An  
7 **System.ComponentModel.ITypeDescriptorContext** that provides a format  
8 context, which can be used to extract additional information about the  
9 environment this converter is being invoked from. This parameter or properties  
10 of this parameter can be **null**. A **System.Globalization.CultureInfo** object to  
11 provide locale information. The object to convert. The type to convert the object  
12 to.

#### 11 *h) GetStandardValues*

13 [C#] public override StandardValuesCollection  
14 GetStandardValues(ITypeDescriptorContext context);

15 [C++] public: StandardValuesCollection\*  
16 GetStandardValues(ITypeDescriptorContext\* context);

17 [VB] Overrides Public Function GetStandardValues(ByVal context As  
18 ITypeDescriptorContext) As StandardValuesCollection

19 [JScript] public override function GetStandardValues(context :  
20 ITypeDescriptorContext) : StandardValuesCollection; Returns a collection of  
21 standard values for the data type this type converter is designed for.

#### 23 *Description*

24 Returns a collection of standard values for the data type this type converter is  
25 designed for when provided with a format context.

*Return Value:* A

**System.ComponentModel.TypeConverter.StandardValuesCollection** that holds a standard set of valid values, or a **null** if the data type does not support a standard set of values.

Returns a collection of index values for the data type.

An **System.ComponentModel.ITypeDescriptorContext** that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be **null**.

*i) GetStandardValuesExclusive*

```
[C#] public override bool GetStandardValuesExclusive(ITypeDescriptorContext context);
```

```
[C++] public: bool GetStandardValuesExclusive(ITypeDescriptorContext* context);
```

```
[VB] Overrides Public Function GetStandardValuesExclusive(ByVal context As ITypeDescriptorContext) As Boolean
```

```
[JScript] public override function GetStandardValuesExclusive(context : ITypeDescriptorContext) : Boolean;
```

*Description*

Determines if the list of standard values returned from `GetStandardValues` is an exclusive list. If the list is exclusive, then no other values are valid, such as in an enum data type. If the list is not exclusive, then there are other valid values besides the list of standard values `GetStandardValues` provides.

*Return Value:* Always returns **false**.

As implemented in this converter, this method always returns **false**. A formatter context.

*j) GetStandardValuesSupported*

```
[C#] public override bool GetStandardValuesSupported(ITypeDescriptorContext
```

```

1 context);
2 [C++] public: bool GetStandardValuesSupported(ITypeDescriptorContext*
3 context);
4 [VB] Overrides Public Function GetStandardValuesSupported(ByVal context As
5 ITypeDescriptorContext) As Boolean
6 [JScript] public override function GetStandardValuesSupported(context :
7 ITypeDescriptorContext) : Boolean; Returns whether this object supports a
8 standard set of values that can be picked from a list.

```

#### *Description*

Determines if this object supports a standard set of values that can be picked from a list.

*Return Value:* Always returns **true** .

As implemented in this class,

**System.Windows.Forms.ImageIndexConverter.GetStandardValuesSupported(System.ComponentModel.ITypeDescriptorContext)** always returns **true** since this object supports a standard set of values that can be picked from a list. An **System.ComponentModel.ITypeDescriptorContext** that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be **null**.

ImageList class (System.Windows.Forms)

#### *a) ToString*

#### *Description*

Provides methods to manage a collection of **System.Drawing.Image** objects. This class cannot be inherited.

**System.Windows.Forms.ImageList** is typically used by other controls, such as the **System.Windows.Forms.ListView** ,

**System.Windows.Forms.TreeView** , or **System.Windows.Forms.ToolBar**  
 . You can add bitmaps, icons, or meta files to the  
**System.Windows.Forms.ImageList** , and the other controls are able to use  
 the images as they require.

*b) ImageList*

*Example Syntax:*

*c) ToString*

|       |            |              |
|-------|------------|--------------|
| [C#]  | public     | ImageList(); |
| [C++] | public:    | ImageList(); |
| [VB]  | Public Sub | New()        |

[JScript] public function ImageList(); Initializes a new instance of the  
**System.Windows.Forms.ImageList** class.

*Description*

Initializes a new instance of the **System.Windows.Forms.ImageList** class  
 with default values for **System.Windows.Forms.ImageList.ColorDepth** ,  
**System.Windows.Forms.ImageList.ImageSize** , and  
**System.Windows.Forms.ImageList.TransparentColor** .

The following table shows initial property values for an instance of  
**System.Windows.Forms.ImageList** .

*d) ImageList*

*Example Syntax:*

*e) ToString*

|       |            |                       |                |
|-------|------------|-----------------------|----------------|
| [C#]  | public     | ImageList(IContainer  | container);    |
| [C++] | public:    | ImageList(IContainer* | container);    |
| [VB]  | Public Sub | New(ByVal container   | As IContainer) |

1 [JScript] public function ImageList(container : IContainer);

2  
3 *Description*

4 Initializes a new instance of the **System.Windows.Forms.ImageList** class,  
associating it with a container.

5 This constructor adds the image list to the specified container. An object  
6 implementing **System.ComponentModel.IContainer** to associate with this  
instance of **System.Windows.Forms.ImageList**.

7  
8 *f) ColorDepth*

9 *g) ToString*

10  
11 [C#] public ColorDepth ColorDepth {get; set;}

12 [C++] public: \_\_property ColorDepth get\_ColorDepth();public: \_\_property void  
13 set\_ColorDepth(ColorDepth);

14 [VB] Public Property ColorDepth As ColorDepth

15 [JScript] public function get ColorDepth() : ColorDepth;public function set  
16 ColorDepth(ColorDepth);

17  
18 *Description*

19 Gets the color depth of the image list.

20 When you set the color depth to a new value, the  
**System.Windows.Forms.ImageList.Handle** for the image list is recreated.

h) *Container*

i) *DesignMode*

j) *Events*

k) *Handle*

l) *ToString*

*Description*

Gets the handle of the image list object.

This corresponds to a Win32 HIMAGELIST handle. Unless you provided the handle in the constructor, the handle is not created until you need to use it. Getting the handle causes it to be created.

m) *HandleCreated*

n) *ToString*

[C#]            public            bool            HandleCreated            {get;}

[C++]           public:            \_\_property            bool            get\_HandleCreated();

[VB]    Public    ReadOnly    Property    HandleCreated    As    Boolean

[JScript]    public    function    get    HandleCreated()    :    Boolean;

*Description*

Gets a value indicating whether the underlying Win32 handle has been created.

o) *Images*

p) *ToString*

```
[C#]      public      ImageList.ImageCollection      Images      {get;}
[C++]     public:  __property  ImageList.ImageCollection*  get_Images();
[VB]      Public  ReadOnly  Property  Images  As  ImageList.ImageCollection
[JScript] public  function  get  Images()  :  ImageList.ImageCollection;
```

*Description*

Gets the **System.Windows.Forms.ImageList.ImageCollection** for this image list.

If the image collection has not yet been created, it is created when you retrieve this property.

q) *ImageSize*

r) *ToString*

```
[C#]      public      Size      ImageSize      {get;      set;}
[C++]     public:  __property  Size  get_ImageSize();public:  __property  void
set_ImageSize(Size);
[VB]      Public      Property      ImageSize      As      Size
[JScript] public  function  get  ImageSize()  :  Size;public  function  set
ImageSize(Size);
```

*Description*

Gets or sets the size of the images in the image list.

When you set a new image size, the handle is recreated.



1           s)     *ImageStream*

2           t)     *ToString*

3  
4 [C#]     public     ImageListStreamer     ImageStream     {get;     set;}

5 [C++]    public:    \_\_property    ImageListStreamer\*    get\_ImageStream();public:

6    \_\_property                void                set\_ImageStream(ImageListStreamer\*);

7 [VB]     Public     Property     ImageStream     As     ImageListStreamer

8 [JScript] public function get ImageStream() : ImageListStreamer;public function

9 set                               ImageStream(ImageListStreamer);

10  
11 *Description*

12 Gets the handle to the **System.Windows.Forms.ImageListStreamer**  
13 associated with this image list.

14 The **System.Windows.Forms.ImageListStreamer** is the data portion of the  
15 image list.

16           u)     *Site*

17           v)     *TransparentColor*

18           w)     *ToString*

19  
20 *Description*

21 Gets or sets the color to treat as transparent.

22 The transparent color is not rendered when the image is drawn.

## x) *ToString*

### *Description*

Occurs when the **System.Windows.Forms.ImageList.Handle** is recreated.

You can use this event to do special processing when the **System.Windows.Forms.ImageList.Handle** is created by actions such as changing the **System.Windows.Forms.ImageList.ColorDepth** or **System.Windows.Forms.ImageList.ImageSize** . Special processing is recommended when you have supplied the handle for the list.

## y) *Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);  
[C++]      protected:      void      Dispose(bool      disposing);  
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)  
[JScript]      protected      override      function      Dispose(disposing      :      Boolean);
```

### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.ImageList** .

Call **System.Windows.Forms.ImageList.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.ImageList** . The **System.Windows.Forms.ImageList.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.ImageList** in an unusable state. After calling **System.Windows.Forms.ImageList.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.ImageList** so the memory it was occupying can be reclaimed by garbage collection.

## z) *Draw*

```
[C#]      public      void      Draw(Graphics      g,      Point      pt,      int      index);
```

```

1 [C++] public: void Draw(Graphics* g, Point pt, int index);
2 [VB] Public Sub Draw(ByVal g As Graphics, ByVal pt As Point, ByVal index As
3 Integer)
4 [JScript] public function Draw(g : Graphics, pt : Point, index : int); Draws the
5 indicated image.

```

#### *Description*

Draws the image indicated by the specified index on the specified **System.Drawing.Graphics** at the given location. **System.Drawing.Graphics** object to draw into. The location defined by a **System.Drawing.Point** at which to draw the image. Index of the image in the **System.Windows.Forms.ImageList** to draw.

#### *aa) Draw*

```

13 [C#] public void Draw(Graphics g, int x, int y, int index);
14 [C++] public: void Draw(Graphics* g, int x, int y, int index);
15 [VB] Public Sub Draw(ByVal g As Graphics, ByVal x As Integer, ByVal y As
16 Integer, ByVal index As Integer)
17 [JScript] public function Draw(g : Graphics, x : int, y : int, index : int);

```

#### *Description*

Draws the image indicated by the given index on the specified **System.Drawing.Graphics** at the specified location. **System.Drawing.Graphics** object to draw into. Horizontal position at which to draw the image. Vertical position at which to draw the image. Index of the image in the **System.Windows.Forms.ImageList** to draw.

**bb) Draw**

```
[C#] public void Draw(Graphics g, int x, int y, int width, int height, int index);  
[C++] public: void Draw(Graphics* g, int x, int y, int width, int height, int index);  
[VB] Public Sub Draw(ByVal g As Graphics, ByVal x As Integer, ByVal y As  
Integer, ByVal width As Integer, ByVal height As Integer, ByVal index As  
Integer)  
[JScript] public function Draw(g : Graphics, x : int, y : int, width : int, height : int,  
index : int);
```

**Description**

Draw the image indicated by the given index using the location, size and raster op code specified.

The image is stretched or compressed as necessary to fit the bounds provided.

**System.Drawing.Graphics** object to draw into. Horizontal position at which to draw the image. Vertical position at which to draw the image. Width of destination image. Height of destination image. Index of the image in the **System.Windows.Forms.ImageList** to draw.

**cc) ToString**

```
[C#] public override string ToString();  
[C++] public: String* ToString();  
[VB] Overrides Public Function ToString() As String  
[JScript] public override function ToString() : String;
```

**Description**

Returns a string representation for this control.

**Return Value:** String Returns a string representation for this control.

ImageListStreamer class (System.Windows.Forms)

*a) ToString*

*Description*

Provides the data portion of an **System.Windows.Forms.ImageList**.

This is a sealed class, so you can not inherit from it. Also, the constructor is private. So you cannot create a new instance of it.

*b) GetObjectData*

[C#] public void GetObjectData(SerializationInfo si, StreamingContext context);

[C++] public: \_\_sealed void GetObjectData(SerializationInfo\* si,  
StreamingContext context);

[VB] NotOverridable Public Sub GetObjectData(ByVal si As SerializationInfo,  
ByVal context As StreamingContext)

[JScript] public function GetObjectData(si : SerializationInfo, context :  
StreamingContext);

*Description*

ImeMode enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies a value that determines the Input Method Editor (IME) status of an object when the object is selected.

An Input Method Editor (IME) allows users to enter and edit Chinese, Japanese, and Korean characters. The IME is an essential component for writing Chinese, Japanese, and Korean scripts. These writing systems have more characters than can be encoded for a regular keyboard. The IMEs for these languages use sequences of base characters that describe an individual character or group of characters to allow you to enter a larger set of characters. Base characters can be component letters from Hangul syllables, phonetic components for Japanese Kanji characters, or various combinations for Chinese characters.

**b) ToString**

|           |         |       |          |          |
|-----------|---------|-------|----------|----------|
| [C#]      | public  | const | ImeMode  | Alpha;   |
| [C++]     | public: | const | ImeMode  | Alpha;   |
| [VB]      | Public  | Const | Alpha As | ImeMode  |
| [JScript] | public  | var   | Alpha :  | ImeMode; |

**Description**

Alphanumeric single-byte characters(SBC). This setting is valid for Korean and Japanese IME only.

**c) ToString**

|           |         |       |              |            |
|-----------|---------|-------|--------------|------------|
| [C#]      | public  | const | ImeMode      | AlphaFull; |
| [C++]     | public: | const | ImeMode      | AlphaFull; |
| [VB]      | Public  | Const | AlphaFull As | ImeMode    |
| [JScript] | public  | var   | AlphaFull :  | ImeMode;   |

**Description**

Alphanumeric double-byte characters. This setting is valid for Korean and Japanese IME only.

**d) ToString**

|           |         |       |         |            |
|-----------|---------|-------|---------|------------|
| [C#]      | public  | const | ImeMode | Disable;   |
| [C++]     | public: | const | ImeMode | Disable;   |
| [VB]      | Public  | Const | Disable | As ImeMode |
| [JScript] | public  | var   | Disable | : ImeMode; |

**Description**

The IME is disabled. With this setting, the users cannot turn the IME on from the keyboard, and the IME floating window is hidden.

**e) ToString**

|           |         |       |         |            |
|-----------|---------|-------|---------|------------|
| [C#]      | public  | const | ImeMode | Hangul;    |
| [C++]     | public: | const | ImeMode | Hangul;    |
| [VB]      | Public  | Const | Hangul  | As ImeMode |
| [JScript] | public  | var   | Hangul  | : ImeMode; |

**Description**

Hangul SBC. This setting is valid for the Korean IME only.

**f) ToString**

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | ImeMode    | HangulFull; |
| [C++]     | public: | const | ImeMode    | HangulFull; |
| [VB]      | Public  | Const | HangulFull | As ImeMode  |
| [JScript] | public  | var   | HangulFull | : ImeMode;  |

*Description*

Hangul DBC. This setting is valid for the Korean IME only.

**g) ToString**

|           |         |       |             |           |
|-----------|---------|-------|-------------|-----------|
| [C#]      | public  | const | ImeMode     | Hiragana; |
| [C++]     | public: | const | ImeMode     | Hiragana; |
| [VB]      | Public  | Const | Hiragana As | ImeMode   |
| [JScript] | public  | var   | Hiragana :  | ImeMode;  |

*Description*

Hiragana DBC. This setting is valid for the Japanese IME only.

**h) ToString**

|           |         |       |            |          |
|-----------|---------|-------|------------|----------|
| [C#]      | public  | const | ImeMode    | Inherit; |
| [C++]     | public: | const | ImeMode    | Inherit; |
| [VB]      | Public  | Const | Inherit As | ImeMode  |
| [JScript] | public  | var   | Inherit :  | ImeMode; |

*Description*

Inherits the IME mode of the parent control.

**i) ToString**

|       |         |       |         |           |
|-------|---------|-------|---------|-----------|
| [C#]  | public  | const | ImeMode | Katakana; |
| [C++] | public: | const | ImeMode | Katakana; |



1 [VB] Public Const Katakana As ImeMode  
2 [JScript] public var Katakana : ImeMode;

4 *Description*  
5 Katakana DBC. This setting is valid for the Japanese IME only.

6 *j) ToString*

8 [C#] public const ImeMode KatakanaHalf;  
9 [C++] public: const ImeMode KatakanaHalf;  
10 [VB] Public Const KatakanaHalf As ImeMode  
11 [JScript] public var KatakanaHalf : ImeMode;

13 *Description*  
14 Katakana SBC. This setting is valid for the Japanese IME only.

15 *k) ToString*

17 [C#] public const ImeMode NoControl;  
18 [C++] public: const ImeMode NoControl;  
19 [VB] Public Const NoControl As ImeMode  
20 [JScript] public var NoControl : ImeMode;

22 *Description*  
23 None (Default).

***l) ToString***

|           |         |       |     |         |          |
|-----------|---------|-------|-----|---------|----------|
| [C#]      | public  | const |     | ImeMode | Off;     |
| [C++]     | public: | const |     | ImeMode | Off;     |
| [VB]      | Public  | Const | Off | As      | ImeMode  |
| [JScript] | public  | var   | Off | :       | ImeMode; |

***Description***

The IME is off. This mode indicates that the IME is off, meaning that the object behaves the same as English entry mode. This setting is valid for Japanese, Simplified Chinese, and Traditional Chinese IME only.

***m) ToString***

|           |         |       |    |         |          |
|-----------|---------|-------|----|---------|----------|
| [C#]      | public  | const |    | ImeMode | On;      |
| [C++]     | public: | const |    | ImeMode | On;      |
| [VB]      | Public  | Const | On | As      | ImeMode  |
| [JScript] | public  | var   | On | :       | ImeMode; |

***Description***

The IME is on. This value indicates that the IME is on and characters specific to Chinese or Japanese can be entered. This setting is valid for Japanese, Simplified Chinese, and Traditional Chinese IME only.

IMessageFilter interface (System.Windows.Forms)

***a) ToString***

***Description***

Defines a message filter interface.

This interface allows an application to capture a message before it is dispatched to a control or form.

### *b) PreFilterMessage*

[C#]            bool            PreFilterMessage(ref            Message            m);

[C++]            bool            PreFilterMessage(Message\*            m);

[VB]    Function    PreFilterMessage(ByRef m As Message) As Boolean

[JScript]    function    PreFilterMessage(m : Message) : Boolean;

### *Description*

Filters out a message before it is dispatched.

*Return Value:* **true** to filter the message and prevent it from being dispatched;  
**false** to allow the message to continue to the next filter or control.

### *Use*

**System.Windows.Forms IMessageFilter.PreFilterMessage(System.Windows.Forms.Message@)** to filter out a message before it is dispatched to a control or form. For example, to prevent the **System.Windows.Forms.Control.Click** event of a **System.Windows.Forms.Button** control from being dispatched to the control, you implement the **System.Windows.Forms IMessageFilter.PreFilterMessage(System.Windows.Forms.Message@)** method and return a **true** value when the **System.Windows.Forms.Control.Click** message occurs. You can also use this method to perform code work that you might need to do before the message is dispatched. The message to be dispatched. You cannot modify this message.

InputLanguage class (System.Windows.Forms)

### *a) PreFilterMessage*

### *Description*

Provides methods and fields to manage the input language. This class cannot be inherited.

An input language is a culture/keyboard layout pair that determines how the physical keys on a keyboard map or plot to characters in a language.

*b) Culture*

*c) PreFilterMessage*

```
[C#]      public      CultureInfo      Culture      {get;}
[C++]      public:      __property      CultureInfo*      get_Culture();
[VB]      Public      ReadOnly      Property      Culture      As      CultureInfo
[JScript]      public      function      get      Culture()      :      CultureInfo;
```

#### *Description*

Gets the culture of the current input language.

This flag is passed to **System.Windows.Forms.InputLanguage.CurrentInputLanguage** to specify that the input language is for the process, not just for the current thread.

*d) CurrentInputLanguage*

*e) PreFilterMessage*

```
[C#]      public      static      InputLanguage      CurrentInputLanguage      {get; set;}
[C++]      public:      __property      static      InputLanguage*
get_CurrentInputLanguage();public:      __property      static      void
set_CurrentInputLanguage(InputLanguage*);
[VB]      Public      Shared      Property      CurrentInputLanguage      As      InputLanguage
[JScript]      public      static      function      get      CurrentInputLanguage()      :
```

1 InputLanguage;public static function set CurrentInputLanguage(InputLanguage);

2  
3 *Description*

4 Gets or sets the input language for the current thread.

5 *f) DefaultInputLanguage*

6 *g) PreFilterMessage*

7  
8 [C#] public static InputLanguage DefaultInputLanguage {get;}

9 [C++] public: \_\_property static InputLanguage\* get\_DefaultInputLanguage();

10 [VB] Public Shared ReadOnly Property DefaultInputLanguage As InputLanguage

11 [JScript] public static function get DefaultInputLanguage() : InputLanguage;

12  
13 *Description*

14 Gets the default input language for the system.

15 *h) Handle*

16 *i) PreFilterMessage*

17  
18 [C#] public IntPtr Handle {get;}

19 [C++] public: \_\_property IntPtr get\_Handle();

20 [VB] Public ReadOnly Property Handle As IntPtr

21 [JScript] public function get Handle() : IntPtr;

22  
23 *Description*

24 Gets the handle for the input language.

j) *InstalledInputLanguages*

k) *PreFilterMessage*

```
[C#] public static InputLanguageCollection InstalledInputLanguages {get;}
[C++] public: __property static InputLanguageCollection*
get_InstalledInputLanguages();
[VB] Public Shared ReadOnly Property InstalledInputLanguages As
InputLanguageCollection
[JScript] public static function get InstalledInputLanguages() :
InputLanguageCollection;
```

*Description*

Gets a list of all installed input languages.

l) *LayoutName*

m) *PreFilterMessage*

```
[C#] public string LayoutName {get;}
[C++] public: __property String* get_LayoutName();
[VB] Public ReadOnly Property LayoutName As String
[JScript] public function get LayoutName() : String;
```

*Description*

Gets the name of the current keyboard layout as it appears in the regional settings of the operating system on the computer.

*n) Equals*

```
[C#]      public      override      bool      Equals(object      value);
[C++]      public:      bool      Equals(Object*      value);
[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean
[JScript] public override function Equals(value : Object) : Boolean;
```

*Description*

Specifies whether two input languages are equal.

*Return Value:* **true** if the two languages are equal; otherwise, **false** . The language to test for equality.

*o) FromCulture*

```
[C#]      public      static      InputLanguage      FromCulture(CultureInfo      culture);
[C++]      public:      static      InputLanguage*      FromCulture(CultureInfo*      culture);
[VB] Public Shared Function FromCulture(ByVal culture As CultureInfo) As
InputLanguage
[JScript] public static function FromCulture(culture : CultureInfo) :
InputLanguage;
```

*Description*

Returns the input language associated with the specified culture.

*Return Value:* An **System.Windows.Forms.InputLanguage** that represents the previously selected input language. The **System.Globalization.CultureInfo** object that specifies the culture to convert from.

p) *GetHashCode*

```
[C#]      public      override      int      GetHashCode();  
[C++]      public:      int      GetHashCode();  
[VB]      Overrides      Public      Function      GetHashCode()      As      Integer  
[JScript]      public      override      function      GetHashCode()      :      int;
```

*Description*

Returns the hash code for this input language.  
*Return Value:* The hash code for this input language.

InputLanguageChangedEventArgs class (System.Windows.Forms)

a) *ToString*

*Description*

Provides data for the  
**System.Windows.Forms.Form.InputLanguageChanged** event.

You can use the data from the  
**System.Windows.Forms.InputLanguageChangedEventArgs** to make decisions about whether to change Input Method Editors (IME) or swap right-to-left values. You can also change a thread's  
**System.Threading.Thread.CurrentCulture** and  
**System.Threading.Thread.CurrentUICulture** properties so that different resources get picked up.

b) *InputLanguageChangedEventArgs*

*Example Syntax:*



c) *ToString*

```
[C#] public InputLanguageChangedEventArgs(CultureInfo culture, byte charSet);  
[C++] public: InputLanguageChangedEventArgs(CultureInfo* culture, unsigned  
char charSet);  
[VB] Public Sub New(ByVal culture As CultureInfo, ByVal charSet As Byte)  
[JScript] public function InputLanguageChangedEventArgs(culture : CultureInfo,  
charSet : Byte); Initializes a new instance of the  
System.Windows.Forms.InputLanguageChangedEventArgs class.
```

*Description*

Initializes a new instance of the **System.Windows.Forms.InputLanguageChangedEventArgs** class with the specified locale and character set. The locale of the input language. The character set associated with the new input language.

d) *InputLanguageChangedEventArgs*

*Example Syntax:*

e) *ToString*

```
[C#] public InputLanguageChangedEventArgs(InputLanguage inputLanguage,  
byte charSet);  
[C++] public: InputLanguageChangedEventArgs(InputLanguage* inputLanguage,  
unsigned char charSet);  
[VB] Public Sub New(ByVal inputLanguage As InputLanguage, ByVal charSet  
As Byte)  
[JScript] public function InputLanguageChangedEventArgs(inputLanguage :
```

1 InputLanguage, charSet : Byte);

2

3 *Description*

4 Initializes a new instance of the  
 5 **System.Windows.Forms.InputLanguageChangedEventArgs** class with the  
 specified input language and character set.

6 The input language specifies a culture/keyboard layout pair. The input language.  
 The character set associated with the new input language.

7

8 f) *CharSet*

9

g) *ToString*

10

11 [C#] public byte CharSet {get;}

12 [C++] public: \_\_property unsigned char get\_CharSet();

13 [VB] Public ReadOnly Property CharSet As Byte

14 [JScript] public function get CharSet() : Byte;

15

16 *Description*

17 Gets the character set associated with the new input language.

18 This property is the Win32 character set that the user switched to. On ANSI  
 19 systems, this property can be used to create fonts that can display the correct  
 20 character set. On Unicode systems, you typically do not need to use this  
 property. Instead, use the **System.Globalization.CultureInfo** class for these  
 functionalities.

21

h) *Culture*

22

i) *ToString*

23

24 [C#] public CultureInfo Culture {get;}

25 [C++] public: \_\_property CultureInfo\* get\_Culture();

```

1  [VB]      Public      ReadOnly      Property      Culture      As      CultureInfo
2  [JScript]      public      function      get      Culture()      :      CultureInfo;

```

#### *Description*

Gets the locale of the input language.

The **System.Windows.Forms.InputLanguageChangedEventArgs.Culture** property specifies a **System.Globalization.CultureInfo** object defining a set of user preference information dependent on the user's language, sub-language, country/region, and cultural conventions.

j) *InputLanguage*

k) *ToString*

```

12 [C#]      public      InputLanguage      InputLanguage      {get;}
13 [C++]      public:      __property      InputLanguage*      get_InputLanguage();
14 [VB]      Public      ReadOnly      Property      InputLanguage      As      InputLanguage
15 [JScript]      public      function      get      InputLanguage()      :      InputLanguage;

```

#### *Description*

Gets a value indicating the input language.

InputLanguageChangedEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **System.Windows.Forms.Form.InputLanguageChanged** event of a **System.Windows.Forms.Form** . The source of the event. An

**System.Windows.Forms.InputLanguageChangedEventArgs** that contains the event data.

When you create a **System.Windows.Forms.InputLanguageChangedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate.

*InputLanguageChangingEventArgs* class (System.Windows.Forms)

*a) ToString*

### *Description*

Provides data for the **System.Windows.Forms.Form.InputLanguageChanging** event.

You can use the data from the **System.Windows.Forms.InputLanguageChangingEventArgs** to prepare to change Input Method Editors (IME) or swap right-to-left values. You can also change a thread's **System.Threading.Thread.CurrentCulture** and **System.Threading.Thread.CurrentUICulture** properties so that different resources get picked up.

*b) InputLanguageChangingEventArgs*

*Example Syntax:*

*c) ToString*

```
[C#] public InputLanguageChangingEventArgs(CultureInfo culture, bool
sysCharSet);
```

```
[C++] public: InputLanguageChangingEventArgs(CultureInfo* culture, bool
sysCharSet);
```

```
[VB] Public Sub New(ByVal culture As CultureInfo, ByVal sysCharSet As
Boolean)
```

[JScript] public function InputLanguageChangingEventArgs(culture : CultureInfo,  
 sysCharSet : Boolean); Initializes a new instance of the  
**System.Windows.Forms.InputLanguageChangingEventArgs** class.

#### *Description*

Initializes a new instance of the **System.Windows.Forms.InputLanguageChangingEventArgs** class with the specified locale, character set, and acceptance. The locale of the requested input language. **true** if the the system default font supports the character set required for the requested input language; otherwise, **false**.

d) *InputLanguageChangingEventArgs*

*Example Syntax:*

e) *ToString*

[C#] public InputLanguageChangingEventArgs(InputLanguage inputLanguage,  
 bool sysCharSet);

[C++] public: InputLanguageChangingEventArgs(InputLanguage\*  
 inputLanguage, bool sysCharSet);

[VB] Public Sub New(ByVal inputLanguage As InputLanguage, ByVal  
 sysCharSet As Boolean)

[JScript] public function InputLanguageChangingEventArgs(inputLanguage :  
 InputLanguage, sysCharSet : Boolean);

#### *Description*

Initializes a new instance of the **System.Windows.Forms.InputLanguageChangingEventArgs** class with the specified input language, character set, and acceptance of a language

change. The requested input language. **true** if the system default font supports the character set required for the requested input language; otherwise, **false**.

*f) Cancel*

*g) Culture*

*h) ToString*

### *Description*

Gets the locale of the requested input language.

The **System.Windows.Forms.InputLanguageChangedEventArgs.Culture** property specifies a **System.Globalization.CultureInfo** object defining a set of user preference information dependent on the user's language, sublanguage, country/region, and cultural conventions.

*i) InputLanguage*

*j) ToString*

[C#]            public            InputLanguage            InputLanguage            {get;}

[C++]        public:        \_\_property        InputLanguage\*        get\_InputLanguage();

[VB]    Public    ReadOnly    Property    InputLanguage    As    InputLanguage

[JScript]    public    function    get    InputLanguage()    :    InputLanguage;

### *Description*

Gets a value indicating the input language.

k) *SysCharSet*

l) *ToString*

|           |         |            |                  |                       |
|-----------|---------|------------|------------------|-----------------------|
| [C#]      | public  | bool       | SysCharSet       | {get;}                |
| [C++]     | public: | __property | bool             | get_SysCharSet();     |
| [VB]      | Public  | ReadOnly   | Property         | SysCharSet As Boolean |
| [JScript] | public  | function   | get SysCharSet() | : Boolean;            |

*Description*

Gets a value indicating whether the system default font supports the character set required for the requested input language.

InputLanguageChangingEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **System.Windows.Forms.Form.InputLanguageChanging** event of a **System.Windows.Forms.Form**. The source of the event. An **System.Windows.Forms.InputLanguageChangingEventArgs** that contains the event data.

When you create an **System.Windows.Forms.InputLanguageChangingEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate.

1        InputLanguageCollection class (System.Windows.Forms)

2        *a)        ToString*

3  
4  
5        *Description*

6        Stores **System.Windows.Forms.InputLanguage** objects.

7        The **System.Windows.Forms.InputLanguageCollection** class is used by  
8        **System.Windows.Forms.InputLanguage** to provide a list of the input  
      languages installed on the system.

9        *b)        Count*

10       *c)        InnerList*

11       *d)        Item*

12       *e)        ToString*

13  
14  
15       *Description*

16       Gets the entry at the specified index of the  
17       **System.Windows.Forms.InputLanguageCollection** . The zero-based index  
18       of the entry to locate in the collection.

19       *f)        Contains*

20  
21       [C#]        public        bool        Contains(InputLanguage        value);

22       [C++]        public:        bool        Contains(InputLanguage\*        value);

23       [VB]        Public Function Contains(ByVal value As InputLanguage) As Boolean

24       [JScript]   public    function    Contains(value : InputLanguage) : Boolean;



## Description

Gets a value indicating whether the **System.Windows.Forms.InputLanguageCollection** contains the specified **System.Windows.Forms.InputLanguage**.

*Return Value:* **true** if the **System.Windows.Forms.InputLanguage** is contained in the collection; otherwise, **false**. The **System.Windows.Forms.InputLanguage** to locate.

### g) CopyTo

[C#] public void CopyTo(InputLanguage[] array, int index);

[C++] public: void CopyTo(InputLanguage\* array[], int index);

[VB] Public Sub CopyTo(ByVal array() As InputLanguage, ByVal index As Integer)

[JScript] public function CopyTo(array : InputLanguage[], index : int);

## Description

Copies the **System.Windows.Forms.InputLanguageCollection** values to a one-dimensional **System.Array** instance at the specified index. The one-dimensional array that is the destination of the values copied from **System.Windows.Forms.InputLanguageCollection**. The index in the *array* parameter where copying begins.

### h) IndexOf

[C#] public int IndexOf(InputLanguage value);

[C++] public: int IndexOf(InputLanguage\* value);

[VB] Public Function IndexOf(ByVal value As InputLanguage) As Integer

[JScript] public function IndexOf(value : InputLanguage) : int;

1  
2 *Description*

3 Returns the index of a **System.Windows.Forms.InputLanguage** in the  
4 **System.Windows.Forms.InputLanguageCollection** .

5 *Return Value:* The index of the **System.Windows.Forms.InputLanguage** in  
6 **System.Windows.Forms.InputLanguageCollection** , if found; otherwise, -  
7 1. The **System.Windows.Forms.InputLanguage** to locate.

8  
9  
10 InvalidateEventArgs class (System.Windows.Forms)

11  
12 a) *ToString*

13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
*Description*

11 Provides data for the  
12 **System.Windows.Forms.Control.Invalidate(System.Drawing.Region)**  
13 event.

14 The  
15 **System.Windows.Forms.Control.Invalidate(System.Drawing.Region)**  
16 event occurs when a control's display is updated. An  
17 **System.Windows.Forms.InvalidateEventArgs** specifies the  
18 **System.Drawing.Graphics** object to use to paint the control and the rectangle  
19 in which to paint.

20  
21 b) *InvalidateEventArgs*

22  
23 *Example Syntax:*

24  
25 c) *ToString*

[C#] public InvalidateEventArgs(Rectangle invalidRect);

[C++] public: InvalidateEventArgs(Rectangle invalidRect);

[VB] Public Sub New(ByVal invalidRect As Rectangle)

[JScript] public function InvalidateEventArgs(invalidRect : Rectangle);

*Description*

Initializes a new instance of the **System.Windows.Forms.InvalidateEventArgs** class. The **System.Drawing.Rectangle** that contains the invalidated window area.

d) *InvalidRect*

e) *ToString*

|           |         |                   |                   |                    |
|-----------|---------|-------------------|-------------------|--------------------|
| [C#]      | public  | Rectangle         | InvalidRect       | {get;}             |
| [C++]     | public: | __property        | Rectangle         | get_InvalidRect(); |
| [VB]      | Public  | ReadOnly Property | InvalidRect       | As Rectangle       |
| [JScript] | public  | function          | get InvalidRect() | : Rectangle;       |

*Description*

Gets the **System.Drawing.Rectangle** that contains the invalidated window area.

InvalidateEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that will handle the **System.Windows.Forms.Control.Invalidate(System.Drawing.Region)** event of a **System.Windows.Forms.Control**. The source of the event. An **System.Windows.Forms.InvalidateEventArgs** that contains the event data.

When you create an **System.Windows.Forms.InvalidateEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The

event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

ItemActivation enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the user action that is required to activate items in a list view control and the feedback that is given as the user moves the mouse pointer over an item.

Use the members of this enumeration to set the value of the **System.Windows.Forms.ListView.Activation** property of the **System.Windows.Forms.ListView** control.

*b) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | ItemActivation | OneClick;         |
| [C++]     | public: | const | ItemActivation | OneClick;         |
| [VB]      | Public  | Const | OneClick       | As ItemActivation |
| [JScript] | public  | var   | OneClick       | : ItemActivation; |

*Description*

The user must single-click to activate items. The cursor changes to a hand pointer cursor, and the item text changes color as the user moves the mouse pointer over the item.

*c) ToString*

|       |         |       |                |           |
|-------|---------|-------|----------------|-----------|
| [C#]  | public  | const | ItemActivation | Standard; |
| [C++] | public: | const | ItemActivation | Standard; |

|           |        |       |          |    |                 |
|-----------|--------|-------|----------|----|-----------------|
| [VB]      | Public | Const | Standard | As | ItemActivation  |
| [JScript] | public | var   | Standard | :  | ItemActivation; |

*Description*

The user must double-click to activate items. No feedback is given as the user moves the mouse pointer over an item.

*d) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | ItemActivation | TwoClick;         |
| [C++]     | public: | const | ItemActivation | TwoClick;         |
| [VB]      | Public  | Const | TwoClick       | As ItemActivation |
| [JScript] | public  | var   | TwoClick       | : ItemActivation; |

*Description*

The user must double-click to activate items and the item text changes color as the user moves the mouse pointer over the item.

ItemBoundsPortion enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies a portion of the list view item from which to retrieve the bounding rectangle.

Use the members of this enumeration when calling the **System.Windows.Forms.ListView.GetItemRect(System.Int32)** method of the **System.Windows.Forms.ListView** control. This enumeration is also used when calling the **System.Windows.Forms.ListViewItem.GetBounds(System.Windows.For**

**ms.ItemBoundsPortion**) method of the **System.Windows.Forms.ListViewItem** class.

*b) ToString*

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C#]      | public  | const | ItemBoundsPortion | Entire;            |
| [C++]     | public: | const | ItemBoundsPortion | Entire;            |
| [VB]      | Public  | Const | Entire As         | ItemBoundsPortion  |
| [JScript] | public  | var   | Entire :          | ItemBoundsPortion; |

*Description*

The bounding rectangle of the entire item, including the icon, the item text, and the subitem text (if displayed), should be retrieved.

*c) ToString*

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C#]      | public  | const | ItemBoundsPortion | Icon;              |
| [C++]     | public: | const | ItemBoundsPortion | Icon;              |
| [VB]      | Public  | Const | Icon As           | ItemBoundsPortion  |
| [JScript] | public  | var   | Icon :            | ItemBoundsPortion; |

*Description*

The bounding rectangle of the icon or small icon should be retrieved.

*d) ToString*

|       |         |       |                   |                   |
|-------|---------|-------|-------------------|-------------------|
| [C#]  | public  | const | ItemBoundsPortion | ItemOnly;         |
| [C++] | public: | const | ItemBoundsPortion | ItemOnly;         |
| [VB]  | Public  | Const | ItemOnly As       | ItemBoundsPortion |

[JScript]      public      var      ItemOnly      :      ItemBoundsPortion;

*Description*

The bounding rectangle of the icon or small icon and the item text should be retrieved. In all views except Report view of the **System.Windows.Forms.ListView**, this value specifies the same bounding rectangle as the **Entire** value. In Report view, this value specifies the bounding rectangle specified by the **Entire** value without the subitems.

*e) ToString*

[C#]              public              const              ItemBoundsPortion              Label;

[C++]              public:              const              ItemBoundsPortion              Label;

[VB]              Public              Const              Label              As              ItemBoundsPortion

[JScript]              public              var              Label              :              ItemBoundsPortion;

*Description*

The bounding rectangle of the item text should be retrieved.

ItemChangedEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **System.Windows.Forms.CurrencyManager.ItemChanged** event.

An **System.Windows.Forms.CurrencyManager.ItemChanged** event occurs whenever the item in a list is changed. For example, this event will occur when the text of the list item is changed to a new value. This event is not raised when the item is moved to a new position within the list because of a new item being added.

b) *Index*

c) *ToString*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | int        | Index    | {get;}           |
| [C++]     | public: | __property | int      | get_Index();     |
| [VB]      | Public  | ReadOnly   | Property | Index As Integer |
| [JScript] | public  | function   | get      | Index() : int;   |

#### *Description*

Indicates the position of the item being changed within the list.

ItemChangedEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **System.Windows.Forms.CurrencyManager.ItemChanged** event of the **System.Windows.Forms.CurrencyManager** class. The source of the event. An **System.Windows.Forms.ItemChangedEventArgs** that contains the event data.

When you create an **System.Windows.Forms.ItemChangedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .



ItemCheckEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **System.Windows.Forms.CheckedListBox.ItemCheck** event of the **System.Windows.Forms.CheckedListBox** and **System.Windows.Forms.ListView** controls.

The **System.Windows.Forms.CheckedListBox.ItemCheck** event occurs when the checked state of an item in a checked list box changes. The **System.Windows.Forms.ItemCheckEventArgs** class specifies the index of the item to change, the current value of the check box for the item, and the new value to set for the check box.

*b) ItemCheckEventArgs*

*Example Syntax:*

*c) ToString*

```
[C#] public ItemCheckEventArgs(int index, CheckState newCheckValue,  
CheckState currentValue);
```

```
[C++] public: ItemCheckEventArgs(int index, CheckState newCheckValue,  
CheckState currentValue);
```

```
[VB] Public Sub New(ByVal index As Integer, ByVal newCheckValue As  
CheckState, ByVal currentValue As CheckState)
```

```
[JScript] public function ItemCheckEventArgs(index : int, newCheckValue :  
CheckState, currentValue : CheckState);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ItemCheckEventArgs** class. The zero-based index of the item to change. One of the **System.Windows.Forms.CheckState** values that indicates whether to change the check box for the item to be checked, unchecked, or indeterminate. One of the **System.Windows.Forms.CheckState** values that indicates whether the check box for the item is currently checked, unchecked, or indeterminate.

d) *CurrentValue*

e) *ToString*

[C#]            public            CheckState            CurrentValue            {get;}

[C++]           public:            \_\_property            CheckState            get\_CurrentValue();

[VB]           Public            ReadOnly            Property            CurrentValue            As            CheckState

[JScript]       public            function            get            CurrentValue()            :            CheckState;

### Description

Gets a value indicating the current state of the item's check box.

This property enables you to determine the check state of the specified item in the **System.Windows.Forms.CheckedListBox** before the check state change to apply is made.

f) *Index*

g) *ToString*

[C#]            public            int            Index            {get;}

[C++]           public:            \_\_property            int            get\_Index();

[VB]           Public            ReadOnly            Property            Index            As            Integer

[JScript]       public            function            get            Index()            :            int;

## Description

Gets the zero-based index of the item to change.

You can use this property to determine which item's check box in the **System.Windows.Forms.CheckedListBox** is being changed.

*h) NewValue*

*i) ToString*

```
[C#] public CheckState NewValue {get; set;}
```

```
[C++] public: __property CheckState get_NewValue();public: __property void  
set_NewValue(CheckState);
```

```
[VB] Public Property NewValue As CheckState
```

```
[JScript] public function get NewValue() : CheckState;public function set  
NewValue(CheckState);
```

## Description

Gets or sets a value indicating whether to set the check box for the item to be checked, unchecked, or indeterminate.

This property enables you to determine the new check state for the specified item before the check state is changed by the **System.Windows.Forms.CheckedListBox** control. In addition to determining the new check state, you can use this property in an event handler for the **System.Windows.Forms.CheckedListBox.ItemCheck** event to change the state to a different check state than the one specified. For example, if the user placed a check mark next to an item in the **System.Windows.Forms.CheckedListBox** that you have determined should not be checked based on the state of your application, you can override the change in the check mark state by setting this property to its previous setting or to a different check state.

ItemCheckEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **ItemCheck** event of a **System.Windows.Forms.CheckedListBox** or **System.Windows.Forms.ListView** control. The source of the event. An **System.Windows.Forms.ItemCheckEventArgs** that contains the event data.

When you create an **System.Windows.Forms.ItemCheckEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ItemDragEventArgs class (System.Windows.Forms)

a) *ToString*

#### *Description*

Provides data for the **System.Windows.Forms.ListView.ItemDrag** event of the **System.Windows.Forms.ListView** and **System.Windows.Forms.TreeView** controls.

The **System.Windows.Forms.ListView.ItemDrag** event occurs when the user begins dragging an item. An **System.Windows.Forms.ItemDragEventArgs** object specifies which mouse button was pressed.

b) *ItemDragEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#]      public      ItemDragEventArgs(MouseButtons      button);  
[C++]     public:      ItemDragEventArgs(MouseButtons      button);  
[VB]      Public      Sub      New(ByVal      button      As      MouseButtons)  
[JScript] public function ItemDragEventArgs(button : MouseButtons); Initializes a  
new instance of the System.Windows.Forms.ItemDragEventArgs class.
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ItemDragEventArgs** class with a specified mouse button.

When this version of the constructor is used, the value of the **System.Windows.Forms.ItemDragEventArgs.Item** property is automatically set to **null** . One of the **System.Windows.Forms.MouseButtons** values that represents which mouse button was pressed.

d) *ItemDragEventArgs*

*Example Syntax:*

e) *ToString*

```
[C#]      public      ItemDragEventArgs(MouseButtons      button,      object      item);  
[C++]     public:      ItemDragEventArgs(MouseButtons      button,      Object*      item);  
[VB]      Public      Sub      New(ByVal      button      As      MouseButtons, ByVal      item      As      Object)  
[JScript] public function ItemDragEventArgs(button : MouseButtons, item :  
Object);
```

*Description*

Initializes a new instance of the

**System.Windows.Forms.ItemDragEventArgs** class with a specified mouse button and the item that is being dragged. One of the **System.Windows.Forms.MouseButtons** values that represents which mouse button was pressed. The item being dragged.

f) *Button*

g) *ToString*

[C#]            public            MouseButtons            Button            {get;}

[C++]           public:            \_\_property            MouseButtons            get\_Button();

[VB]           Public           ReadOnly           Property           Button           As           MouseButtons

[JScript]       public           function           get           Button()           :           MouseButtons;

### *Description*

Gets the name of the mouse button that was clicked during the drag operation.

This property enables you to determine which mouse button was pressed during a drag and drop operation. The value of this property can be used to properly determine how the drag and drop operation should be performed.

h) *Item*

i) *ToString*

[C#]            public            object            Item            {get;}

[C++]           public:            \_\_property            Object\*            get\_Item();

[VB]           Public           ReadOnly           Property           Item           As           Object

[JScript]       public           function           get           Item()           :           Object;

### *Description*

Gets the item that is being dragged.

1 You can use this property to determine which item from the  
2 **System.Windows.Forms.TreeView** or **System.Windows.Forms.ListView**  
controls is being dragged from the control.

3 ItemDragEventHandler delegate (System.Windows.Forms)

4 a) *ToString*

5  
6  
7 *Description*

8 Represents the method that will handle the  
9 **System.Windows.Forms.ListView.ItemDrag** event of a  
10 **System.Windows.Forms.ListView** or **System.Windows.Forms.TreeView**  
control. The source of the event. An  
11 **System.Windows.Forms.ItemDragEventArgs** that contains the event data.

12 When you create an **System.Windows.Forms.ItemDragEventHandler**  
13 delegate, you identify the method that will handle the event. To associate the  
14 event with your event handler, add an instance of the delegate to the event. The  
15 event handler is called whenever the event occurs, unless you remove the  
16 delegate. For more information about event handler delegates, see .

17 IWin32Window interface (System.Windows.Forms)

18 a) *ToString*

19  
20  
21 *Description*

22 Provides an interface to expose Win32 HWND handles.

23 This interface is implemented on objects that expose Win32 HWND handles. The  
24 resultant handle can be used with Win32 API calls.

25 b) *Handle*

c) *ToString*

|      |        |        |        |
|------|--------|--------|--------|
| [C#] | IntPtr | Handle | {get;} |
|------|--------|--------|--------|

|           |          |          |        |          |   |               |
|-----------|----------|----------|--------|----------|---|---------------|
| [C++]     |          |          | IntPtr |          |   | get_Handle(); |
| [VB]      | ReadOnly | Property | Handle | As       |   | IntPtr        |
| [JScript] | abstract | function | get    | Handle() | : | IntPtr;       |

#### Description

Gets the handle to the window represented by the implementer.

Depending on the implementer, the value of the **System.Windows.Forms.IWin32Window.Handle** property could change during the life of the window.

IWindowTarget interface (System.Windows.Forms)

#### a) ToString

#### Description

This interface defines the communication layer between a Control object and the Win32 API. Each Control object has an internal implementation this interface that is called by the Win32 window.

#### b) OnHandleChange

|           |          |                          |           |             |
|-----------|----------|--------------------------|-----------|-------------|
| [C#]      | void     | OnHandleChange(IntPtr    |           | newHandle); |
| [C++]     | void     | OnHandleChange(IntPtr    |           | newHandle); |
| [VB]      | Sub      | OnHandleChange(ByVal     | newHandle | As IntPtr)  |
| [JScript] | function | OnHandleChange(newHandle | :         | IntPtr);    |

#### Description

Called when the window handle of the control has changed. The new window handle value.



c) *OnMessage*

```
[C#]          void          OnMessage(ref          Message          m);
[C++]          void          OnMessage(Message*          m);
[VB]          Sub          OnMessage(ByRef          m          As          Message)
[JScript]          function          OnMessage(m          :          Message);
```

*Description*

Called to do control-specific processing for this window. The message to process.

KeyEventArgs class (System.Windows.Forms)

a) *OnMessage*

*Description*

Provides data for the **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

A **System.Windows.Forms.KeyEventArgs** , which specifies the key the user pressed and whether any modifier keys (CTRL, ALT, and SHIFT) were pressed at the same time, is passed with each **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

b) *KeyEventArgs*

*Example Syntax:*

c) *OnMessage*

```
[C#]          public          KeyEventArgs(Keys          keyData);
[C++]          public:          KeyEventArgs(Keys          keyData);
```

1 [VB] Public Sub New(ByVal keyData As Keys)

2 [JScript] public function KeyEventArgs(keyData : Keys);

4 *Description*

5 Initializes a new instance of the **System.Windows.Forms.KeyEventArgs**  
6 class. The key code constant for the key that was pressed, combined with any  
7 modifier flags that indicate which CTRL, SHIFT, and ALT keys were pressed at  
the same time. Possible values are obtained by applying the bitwise OR (|)  
operator to constants from **System.Windows.Forms.Keys**.

8 *d) Alt*

9 *e) OnMessage*

11 [C#] public virtual bool Alt {get;}

12 [C++] public: \_\_property virtual bool get\_Alt();

13 [VB] Overridable Public ReadOnly Property Alt As Boolean

14 [JScript] public function get Alt() : Boolean;

16 *Description*

17 Gets a value indicating whether the ALT key was pressed.

18 *f) Control*

19 *g) OnMessage*

21 [C#] public bool Control {get;}

22 [C++] public: \_\_property bool get\_Control();

23 [VB] Public ReadOnly Property Control As Boolean

24 [JScript] public function get Control() : Boolean;

*Description*

Gets a value indicating whether the CTRL key was pressed.

*h) Handled*

*i) OnMessage*

[C#] public bool Handled {get; set;}

[C++] public: \_\_property bool get\_Handled();public: \_\_property void  
set\_Handled(bool);

[VB] Public Property Handled As Boolean

[JScript] public function get Handled() : Boolean;public function set  
Handled(Boolean);

*Description*

Gets or sets a value indicating whether the event was handled.

*j) KeyCode*

*k) OnMessage*

[C#] public Keys KeyCode {get;}

[C++] public: \_\_property Keys get\_KeyCode();

[VB] Public ReadOnly Property KeyCode As Keys

[JScript] public function get KeyCode() : Keys;

*Description*

Gets the keyboard code for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

l) *KeyData*

m) *OnMessage*

|           |         |            |          |                   |
|-----------|---------|------------|----------|-------------------|
| [C#]      | public  | Keys       | KeyData  | {get;}            |
| [C++]     | public: | __property | Keys     | get_KeyData();    |
| [VB]      | Public  | ReadOnly   | Property | KeyData As Keys   |
| [JScript] | public  | function   | get      | KeyData() : Keys; |

#### *Description*

Gets the key data for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

You can use constants from **System.Windows.Forms.Keys** to extract information from the **System.Windows.Forms.KeyEventArgs.KeyData** property. Use the bitwise AND (&) operator to compare data returned by **System.Windows.Forms.KeyEventArgs.KeyData** with constants in **System.Windows.Forms.Keys** to obtain information about which keys the user pressed. To determine whether a specific modifier key was pressed, use the **System.Windows.Forms.KeyEventArgs.Control** , **System.Windows.Forms.KeyEventArgs.Shift** , and **System.Windows.Forms.KeyEventArgs.Alt** properties.

n) *KeyValue*

o) *OnMessage*

|           |         |            |          |                     |
|-----------|---------|------------|----------|---------------------|
| [C#]      | public  | int        | KeyValue | {get;}              |
| [C++]     | public: | __property | int      | get_KeyValue();     |
| [VB]      | Public  | ReadOnly   | Property | KeyValue As Integer |
| [JScript] | public  | function   | get      | KeyValue() : int;   |

## Description

Gets the keyboard value for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

*p) Modifiers*

*q) OnMessage*

|           |         |            |           |                     |
|-----------|---------|------------|-----------|---------------------|
| [C#]      | public  | Keys       | Modifiers | {get;}              |
| [C++]     | public: | __property | Keys      | get_Modifiers();    |
| [VB]      | Public  | ReadOnly   | Property  | Modifiers As Keys   |
| [JScript] | public  | function   | get       | Modifiers() : Keys; |

## Description

Gets the modifier flags for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event. This indicates which modifier keys (CTRL, SHIFT, and/or ALT) were pressed.

To determine whether a specific modifier key was pressed, use the **System.Windows.Forms.KeyEventArgs.Control**, **System.Windows.Forms.KeyEventArgs.Shift**, and **System.Windows.Forms.KeyEventArgs.Alt** properties. Modifier flags can be combined with bitwise OR.

*r) Shift*

*s) OnMessage*

|           |             |            |          |          |                  |
|-----------|-------------|------------|----------|----------|------------------|
| [C#]      | public      | virtual    | bool     | Shift    | {get;}           |
| [C++]     | public:     | __property | virtual  | bool     | get_Shift();     |
| [VB]      | Overridable | Public     | ReadOnly | Property | Shift As Boolean |
| [JScript] | public      | function   | get      | Shift()  | : Boolean;       |

1  
2 *Description*

3 Gets a value indicating whether the SHIFT key was pressed.

4       KeyEventHandler delegate (System.Windows.Forms)

5       a)     *ToString*

6  
7  
8 *Description*

9 Represents a method that will handle the  
10 **System.Windows.Forms.Control.KeyUp** or  
11 **System.Windows.Forms.Control.KeyDown** event of a  
12 **System.Windows.Forms.Control** . The source of the event. A  
13 **System.Windows.Forms.KeyEventArgs** that contains the event data.

14 When you create a **System.Windows.Forms.KeyEventHandler** delegate, you  
15 identify the method that will handle the event. To associate the event with your  
16 event handler, add an instance of the delegate to the event. The event handler is  
17 called whenever the event occurs, unless you remove the delegate. For more  
18 information about handling events with delegates, see .

19       KeyPressEventArgs class (System.Windows.Forms)

20       a)     *ToString*

21  
22  
23  
24  
25 *Description*

Provides data for the **System.Windows.Forms.Control.KeyPress** event.

A **System.Windows.Forms.KeyPressEventArgs** specifies the character that  
is composed when the user presses a key. For example, when the user presses  
SHIFT + K, the **System.Windows.Forms.KeyPressEventArgs.KeyChar**  
property returns an uppercase K.

b)     *KeyPressEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#]      public      KeyPressEventArgs(char      keyChar);
[C++]     public:      KeyPressEventArgs(__wchar_t      keyChar);
[VB]      Public      Sub      New(ByVal      keyChar      As      Char)
[JScript] public      function      KeyPressEventArgs(keyChar      :      Char);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.KeyPressEventArgs** class. The ASCII character corresponding to the key the user pressed.

d) *Handled*

e) *ToString*

```
[C#]      public      bool      Handled      {get;      set;}
[C++]     public:      __property      bool      get_Handled();public:      __property      void
set_Handled(bool);
[VB]      Public      Property      Handled      As      Boolean
[JScript] public      function      get      Handled()      :      Boolean;public      function      set
Handled(Boolean);
```

*Description*

Gets or sets a value indicating whether the **System.Windows.Forms.Control.KeyPress** event was handled.

If the event is not handled, it will be sent to Windows for default processing.

f) *KeyChar*

g) *ToString*

|           |         |            |               |                 |
|-----------|---------|------------|---------------|-----------------|
| [C#]      | public  | char       | KeyChar       | {get;}          |
| [C++]     | public: | __property | __wchar_t     | get_KeyChar();  |
| [VB]      | Public  | ReadOnly   | Property      | KeyChar As Char |
| [JScript] | public  | function   | get KeyChar() | : Char;         |

#### *Description*

Gets the character corresponding to the key pressed.

KeyPressEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents a method that will handle the **System.Windows.Forms.Control.KeyPress** event of a **System.Windows.Forms.Control** . The source of the event. A **System.Windows.Forms.KeyPressEventArgs** that contains the event data.

When you create a **System.Windows.Forms.KeyPressEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .



## Keys enumeration (System.Windows.Forms)

### a) *ToString*

#### *Description*

Specifies key codes and modifiers.

This class contains constants to use for processing keyboard input. Keys are identified by key values, which consist of a key code and a set of modifiers combined into a single integer value. The four left digits of a key value contain the key code (which is the same as a Windows virtual key code). The four right digits of a key value contain modifier bits for the SHIFT, CONTROL, and ALT keys.

### b) *ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | A;      |
| [C++]     | public: | const | Keys | A;      |
| [VB]      | Public  | Const | A    | As Keys |
| [JScript] | public  | var   | A    | : Keys; |

#### *Description*

The A key.

### c) *ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Add;    |
| [C++]     | public: | const | Keys | Add;    |
| [VB]      | Public  | Const | Add  | As Keys |
| [JScript] | public  | var   | Add  | : Keys; |

*Description*

The Add key.

*d) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Alt;    |
| [C++]     | public: | const | Keys | Alt;    |
| [VB]      | Public  | Const | Alt  | As Keys |
| [JScript] | public  | var   | Alt  | : Keys; |

*Description*

The ALT modifier key.

*e) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Apps;   |
| [C++]     | public: | const | Keys | Apps;   |
| [VB]      | Public  | Const | Apps | As Keys |
| [JScript] | public  | var   | Apps | : Keys; |

*Description*

The Application key (Microsoft Natural Keyboard).

*f) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Attn; |
| [C++] | public: | const | Keys | Attn; |

|           |        |       |      |    |       |
|-----------|--------|-------|------|----|-------|
| [VB]      | Public | Const | Attn | As | Keys  |
| [JScript] | public | var   | Attn | :  | Keys; |

*Description*

The ATTN key.

*g) ToString*

|           |         |       |   |      |       |
|-----------|---------|-------|---|------|-------|
| [C#]      | public  | const |   | Keys | B;    |
| [C++]     | public: | const |   | Keys | B;    |
| [VB]      | Public  | Const | B | As   | Keys  |
| [JScript] | public  | var   | B | :    | Keys; |

*Description*

The B key.

*h) ToString*

|           |         |       |      |      |       |
|-----------|---------|-------|------|------|-------|
| [C#]      | public  | const |      | Keys | Back; |
| [C++]     | public: | const |      | Keys | Back; |
| [VB]      | Public  | Const | Back | As   | Keys  |
| [JScript] | public  | var   | Back | :    | Keys; |

*Description*

The BACKSPACE key.

**i) ToString**

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | BrowserBack; |
| [C++]     | public: | const | Keys        | BrowserBack; |
| [VB]      | Public  | Const | BrowserBack | As Keys      |
| [JScript] | public  | var   | BrowserBack | : Keys;      |

*Description*

The Browser Back key (Windows 2000 or later).

**j) ToString**

|           |         |       |                  |                   |
|-----------|---------|-------|------------------|-------------------|
| [C#]      | public  | const | Keys             | BrowserFavorites; |
| [C++]     | public: | const | Keys             | BrowserFavorites; |
| [VB]      | Public  | Const | BrowserFavorites | As Keys           |
| [JScript] | public  | var   | BrowserFavorites | : Keys;           |

*Description*

The Browser Favorites key (Windows 2000 or later).

*Description*

The Browser Favorites key.

**k) ToString**

|       |         |       |                |                 |
|-------|---------|-------|----------------|-----------------|
| [C#]  | public  | const | Keys           | BrowserForward; |
| [C++] | public: | const | Keys           | BrowserForward; |
| [VB]  | Public  | Const | BrowserForward | As Keys         |

1 [JScript] public var BrowserForward : Keys;

3 *Description*

4 The Browser Forward key (Windows 2000 or later).

5 *l) ToString*

7 [C#] public const Keys BrowserHome;

8 [C++] public: const Keys BrowserHome;

9 [VB] Public Const BrowserHome As Keys

10 [JScript] public var BrowserHome : Keys;

12 *Description*

13 The Browser Home key (Windows 2000 or later).

14 *m) ToString*

16 [C#] public const Keys BrowserRefresh;

17 [C++] public: const Keys BrowserRefresh;

18 [VB] Public Const BrowserRefresh As Keys

19 [JScript] public var BrowserRefresh : Keys;

21 *Description*

22 The Browser Refresh key (Windows 2000 or later).

23 *n) ToString*

25 [C#] public const Keys BrowserSearch;

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C++]     | public: | const | Keys          | BrowserSearch; |
| [VB]      | Public  | Const | BrowserSearch | As Keys        |
| [JScript] | public  | var   | BrowserSearch | : Keys;        |

*Description*

The Browser Search key (Windows 2000 or later).

*o) ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | BrowserStop; |
| [C++]     | public: | const | Keys        | BrowserStop; |
| [VB]      | Public  | Const | BrowserStop | As Keys      |
| [JScript] | public  | var   | BrowserStop | : Keys;      |

*Description*

The Browser Stop key (Windows 2000 or later).

*p) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | C;      |
| [C++]     | public: | const | Keys | C;      |
| [VB]      | Public  | Const | C    | As Keys |
| [JScript] | public  | var   | C    | : Keys; |

*Description*

The C key.

**q) ToString**

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Cancel; |
| [C++]     | public: | const | Keys   | Cancel; |
| [VB]      | Public  | Const | Cancel | As Keys |
| [JScript] | public  | var   | Cancel | : Keys; |

*Description*

The CANCEL key.

**r) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Capital; |
| [C++]     | public: | const | Keys    | Capital; |
| [VB]      | Public  | Const | Capital | As Keys  |
| [JScript] | public  | var   | Capital | : Keys;  |

*Description*

The CAPS LOCK key.

**s) ToString**

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | CapsLock; |
| [C++]     | public: | const | Keys     | CapsLock; |
| [VB]      | Public  | Const | CapsLock | As Keys   |
| [JScript] | public  | var   | CapsLock | : Keys;   |

*Description*

The CAPS LOCK key.

*t) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Clear;  |
| [C++]     | public: | const | Keys  | Clear;  |
| [VB]      | Public  | Const | Clear | As Keys |
| [JScript] | public  | var   | Clear | : Keys; |

*Description*

The CLEAR key.

*u) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Control; |
| [C++]     | public: | const | Keys    | Control; |
| [VB]      | Public  | Const | Control | As Keys  |
| [JScript] | public  | var   | Control | : Keys;  |

*Description*

The CTRL modifier key.

*v) ToString*

|       |         |       |      |             |
|-------|---------|-------|------|-------------|
| [C#]  | public  | const | Keys | ControlKey; |
| [C++] | public: | const | Keys | ControlKey; |



1 [VB] Public Const ControlKey As Keys  
 2 [JScript] public var ControlKey : Keys;

4 *Description*  
 5 The CTRL key.

6 w) *ToString*

8 [C#] public const Keys Crsel;  
 9 [C++] public: const Keys Crsel;  
 10 [VB] Public Const Crsel As Keys  
 11 [JScript] public var Crsel : Keys;

13 *Description*  
 14 The CRSEL key.

15 x) *ToString*

17 [C#] public const Keys D;  
 18 [C++] public: const Keys D;  
 19 [VB] Public Const D As Keys  
 20 [JScript] public var D : Keys;

22 *Description*  
 23 The D key.

y) *ToString*

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | D0;   |
| [C++]     | public: | const |    | Keys | D0;   |
| [VB]      | Public  | Const | D0 | As   | Keys  |
| [JScript] | public  | var   | D0 | :    | Keys; |

*Description*

The 0 key.

z) *ToString*

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | D1;   |
| [C++]     | public: | const |    | Keys | D1;   |
| [VB]      | Public  | Const | D1 | As   | Keys  |
| [JScript] | public  | var   | D1 | :    | Keys; |

*Description*

The 1 key.

aa) *ToString*

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | D2;   |
| [C++]     | public: | const |    | Keys | D2;   |
| [VB]      | Public  | Const | D2 | As   | Keys  |
| [JScript] | public  | var   | D2 | :    | Keys; |

*Description*

The 2 key.

***bb) ToString***

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | D3;   |
| [C++]     | public: | const |    | Keys | D3;   |
| [VB]      | Public  | Const | D3 | As   | Keys  |
| [JScript] | public  | var   | D3 | :    | Keys; |

*Description*

The 3 key.

***cc) ToString***

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | D4;   |
| [C++]     | public: | const |    | Keys | D4;   |
| [VB]      | Public  | Const | D4 | As   | Keys  |
| [JScript] | public  | var   | D4 | :    | Keys; |

*Description*

The 4 key.

***dd) ToString***

|       |         |       |  |      |     |
|-------|---------|-------|--|------|-----|
| [C#]  | public  | const |  | Keys | D5; |
| [C++] | public: | const |  | Keys | D5; |

|           |        |       |    |    |       |
|-----------|--------|-------|----|----|-------|
| [VB]      | Public | Const | D5 | As | Keys  |
| [JScript] | public | var   | D5 | :  | Keys; |

*Description*

The 5 key.

*ee) ToString*

|      |        |       |  |      |     |
|------|--------|-------|--|------|-----|
| [C#] | public | const |  | Keys | D6; |
|------|--------|-------|--|------|-----|

|       |         |       |  |      |     |
|-------|---------|-------|--|------|-----|
| [C++] | public: | const |  | Keys | D6; |
|-------|---------|-------|--|------|-----|

|      |        |       |    |    |      |
|------|--------|-------|----|----|------|
| [VB] | Public | Const | D6 | As | Keys |
|------|--------|-------|----|----|------|

|           |        |     |    |   |       |
|-----------|--------|-----|----|---|-------|
| [JScript] | public | var | D6 | : | Keys; |
|-----------|--------|-----|----|---|-------|

*Description*

The 6 key.

*ff) ToString*

|      |        |       |  |      |     |
|------|--------|-------|--|------|-----|
| [C#] | public | const |  | Keys | D7; |
|------|--------|-------|--|------|-----|

|       |         |       |  |      |     |
|-------|---------|-------|--|------|-----|
| [C++] | public: | const |  | Keys | D7; |
|-------|---------|-------|--|------|-----|

|      |        |       |    |    |      |
|------|--------|-------|----|----|------|
| [VB] | Public | Const | D7 | As | Keys |
|------|--------|-------|----|----|------|

|           |        |     |    |   |       |
|-----------|--------|-----|----|---|-------|
| [JScript] | public | var | D7 | : | Keys; |
|-----------|--------|-----|----|---|-------|

*Description*

The 7 key.

**gg) ToString**

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D8;   |
| [C++]     | public: | const | Keys  | D8;   |
| [VB]      | Public  | Const | D8 As | Keys  |
| [JScript] | public  | var   | D8 :  | Keys; |

*Description*

The 8 key.

**hh) ToString**

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D9;   |
| [C++]     | public: | const | Keys  | D9;   |
| [VB]      | Public  | Const | D9 As | Keys  |
| [JScript] | public  | var   | D9 :  | Keys; |

*Description*

The 9 key.

**ii) ToString**

|           |         |       |            |          |
|-----------|---------|-------|------------|----------|
| [C#]      | public  | const | Keys       | Decimal; |
| [C++]     | public: | const | Keys       | Decimal; |
| [VB]      | Public  | Const | Decimal As | Keys     |
| [JScript] | public  | var   | Decimal :  | Keys;    |

*Description*

The Decimal key.

*jj) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Delete; |
| [C++]     | public: | const | Keys   | Delete; |
| [VB]      | Public  | Const | Delete | As Keys |
| [JScript] | public  | var   | Delete | : Keys; |

*Description*

The DEL key.

*kk) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Divide; |
| [C++]     | public: | const | Keys   | Divide; |
| [VB]      | Public  | Const | Divide | As Keys |
| [JScript] | public  | var   | Divide | : Keys; |

*Description*

The Divide key.

*ll) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Down; |
| [C++] | public: | const | Keys | Down; |

|    |                     |         |       |      |      |       |
|----|---------------------|---------|-------|------|------|-------|
| 1  | [VB]                | Public  | Const | Down | As   | Keys  |
| 2  | [JScript]           | public  | var   | Down | :    | Keys; |
| 3  |                     |         |       |      |      |       |
| 4  | <i>Description</i>  |         |       |      |      |       |
| 5  | The DOWN ARROW key. |         |       |      |      |       |
| 6  | <i>mm) ToString</i> |         |       |      |      |       |
| 7  |                     |         |       |      |      |       |
| 8  | [C#]                | public  | const |      | Keys | E;    |
| 9  | [C++]               | public: | const |      | Keys | E;    |
| 10 | [VB]                | Public  | Const | E    | As   | Keys  |
| 11 | [JScript]           | public  | var   | E    | :    | Keys; |
| 12 |                     |         |       |      |      |       |
| 13 | <i>Description</i>  |         |       |      |      |       |
| 14 | The E key.          |         |       |      |      |       |
| 15 | <i>nn) ToString</i> |         |       |      |      |       |
| 16 |                     |         |       |      |      |       |
| 17 | [C#]                | public  | const |      | Keys | End;  |
| 18 | [C++]               | public: | const |      | Keys | End;  |
| 19 | [VB]                | Public  | Const | End  | As   | Keys  |
| 20 | [JScript]           | public  | var   | End  | :    | Keys; |
| 21 |                     |         |       |      |      |       |
| 22 | <i>Description</i>  |         |       |      |      |       |
| 23 | The END key.        |         |       |      |      |       |
| 24 |                     |         |       |      |      |       |
| 25 |                     |         |       |      |      |       |

*oo) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Enter;  |
| [C++]     | public: | const | Keys  | Enter;  |
| [VB]      | Public  | Const | Enter | As Keys |
| [JScript] | public  | var   | Enter | : Keys; |

*Description*

The ENTER key.

*pp) ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | EraseEof; |
| [C++]     | public: | const | Keys     | EraseEof; |
| [VB]      | Public  | Const | EraseEof | As Keys   |
| [JScript] | public  | var   | EraseEof | : Keys;   |

*Description*

The ERASE EOF key.

*qq) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Escape; |
| [C++]     | public: | const | Keys   | Escape; |
| [VB]      | Public  | Const | Escape | As Keys |
| [JScript] | public  | var   | Escape | : Keys; |



*Description*

The ESC key.

*rr) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Execute; |
| [C++]     | public: | const | Keys    | Execute; |
| [VB]      | Public  | Const | Execute | As Keys  |
| [JScript] | public  | var   | Execute | : Keys;  |

*Description*

The EXECUTE key.

*ss) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Exsel;  |
| [C++]     | public: | const | Keys  | Exsel;  |
| [VB]      | Public  | Const | Exsel | As Keys |
| [JScript] | public  | var   | Exsel | : Keys; |

*Description*

The EXSEL key.

*tt) ToString*

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C#]  | public  | const | Keys | F; |
| [C++] | public: | const | Keys | F; |

|   |           |        |       |   |    |       |
|---|-----------|--------|-------|---|----|-------|
| 1 | [VB]      | Public | Const | F | As | Keys  |
| 2 | [JScript] | public | var   | F | :  | Keys; |

3

4 *Description*

5 The F key.

6 *uu) ToString*

|    |           |         |       |    |      |       |
|----|-----------|---------|-------|----|------|-------|
| 8  | [C#]      | public  | const |    | Keys | F1;   |
| 9  | [C++]     | public: | const |    | Keys | F1;   |
| 10 | [VB]      | Public  | Const | F1 | As   | Keys  |
| 11 | [JScript] | public  | var   | F1 | :    | Keys; |

12

13 *Description*

14 The F1 key.

15 *vv) ToString*

|    |           |         |       |     |      |       |
|----|-----------|---------|-------|-----|------|-------|
| 17 | [C#]      | public  | const |     | Keys | F10;  |
| 18 | [C++]     | public: | const |     | Keys | F10;  |
| 19 | [VB]      | Public  | Const | F10 | As   | Keys  |
| 20 | [JScript] | public  | var   | F10 | :    | Keys; |

21

22 *Description*

23 The F10 key.

24

25

*ww) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F11;  |
| [C++]     | public: | const | Keys   | F11;  |
| [VB]      | Public  | Const | F11 As | Keys  |
| [JScript] | public  | var   | F11 :  | Keys; |

*Description*

The F11 key.

*xx) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F12;  |
| [C++]     | public: | const | Keys   | F12;  |
| [VB]      | Public  | Const | F12 As | Keys  |
| [JScript] | public  | var   | F12 :  | Keys; |

*Description*

The F12 key.

*yy) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F13;  |
| [C++]     | public: | const | Keys   | F13;  |
| [VB]      | Public  | Const | F13 As | Keys  |
| [JScript] | public  | var   | F13 :  | Keys; |

*Description*

The F13 key.

*zz) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F14;    |
| [C++]     | public: | const | Keys | F14;    |
| [VB]      | Public  | Const | F14  | As Keys |
| [JScript] | public  | var   | F14  | : Keys; |

*Description*

The F14 key.

*aaa) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F15;    |
| [C++]     | public: | const | Keys | F15;    |
| [VB]      | Public  | Const | F15  | As Keys |
| [JScript] | public  | var   | F15  | : Keys; |

*Description*

The F15 key.

*bbb) ToString*

|       |         |       |      |      |
|-------|---------|-------|------|------|
| [C#]  | public  | const | Keys | F16; |
| [C++] | public: | const | Keys | F16; |

|           |        |       |     |    |       |
|-----------|--------|-------|-----|----|-------|
| [VB]      | Public | Const | F16 | As | Keys  |
| [JScript] | public | var   | F16 | :  | Keys; |

*Description*  
The F16 key.

*ccc) ToString*

|           |         |       |     |      |       |
|-----------|---------|-------|-----|------|-------|
| [C#]      | public  | const |     | Keys | F17;  |
| [C++]     | public: | const |     | Keys | F17;  |
| [VB]      | Public  | Const | F17 | As   | Keys  |
| [JScript] | public  | var   | F17 | :    | Keys; |

*Description*  
The F17 key.

*ddd) ToString*

|           |         |       |     |      |       |
|-----------|---------|-------|-----|------|-------|
| [C#]      | public  | const |     | Keys | F18;  |
| [C++]     | public: | const |     | Keys | F18;  |
| [VB]      | Public  | Const | F18 | As   | Keys  |
| [JScript] | public  | var   | F18 | :    | Keys; |

*Description*  
The F18 key.

*eee) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F19;  |
| [C++]     | public: | const | Keys   | F19;  |
| [VB]      | Public  | Const | F19 As | Keys  |
| [JScript] | public  | var   | F19 :  | Keys; |

*Description*

The F19 key.

*fff) ToString*

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | F2;   |
| [C++]     | public: | const | Keys  | F2;   |
| [VB]      | Public  | Const | F2 As | Keys  |
| [JScript] | public  | var   | F2 :  | Keys; |

*Description*

The F2 key.

*ggg) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F20;  |
| [C++]     | public: | const | Keys   | F20;  |
| [VB]      | Public  | Const | F20 As | Keys  |
| [JScript] | public  | var   | F20 :  | Keys; |

*Description*

The F20 key.

*hhh) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F21;    |
| [C++]     | public: | const | Keys | F21;    |
| [VB]      | Public  | Const | F21  | As Keys |
| [JScript] | public  | var   | F21  | : Keys; |

*Description*

The F21 key.

*iii) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F22;    |
| [C++]     | public: | const | Keys | F22;    |
| [VB]      | Public  | Const | F22  | As Keys |
| [JScript] | public  | var   | F22  | : Keys; |

*Description*

The F22 key.

*jjj) ToString*

|       |         |       |      |      |
|-------|---------|-------|------|------|
| [C#]  | public  | const | Keys | F23; |
| [C++] | public: | const | Keys | F23; |

|           |        |       |     |    |       |
|-----------|--------|-------|-----|----|-------|
| [VB]      | Public | Const | F23 | As | Keys  |
| [JScript] | public | var   | F23 | :  | Keys; |

*Description*  
The F23 key.

*kkk) ToString*

|           |         |       |     |      |       |
|-----------|---------|-------|-----|------|-------|
| [C#]      | public  | const |     | Keys | F24;  |
| [C++]     | public: | const |     | Keys | F24;  |
| [VB]      | Public  | Const | F24 | As   | Keys  |
| [JScript] | public  | var   | F24 | :    | Keys; |

*Description*  
The F24 key.

*lll) ToString*

|           |         |       |    |      |       |
|-----------|---------|-------|----|------|-------|
| [C#]      | public  | const |    | Keys | F3;   |
| [C++]     | public: | const |    | Keys | F3;   |
| [VB]      | Public  | Const | F3 | As   | Keys  |
| [JScript] | public  | var   | F3 | :    | Keys; |

*Description*  
The F3 key.



*mmm) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F4;     |
| [C++]     | public: | const | Keys | F4;     |
| [VB]      | Public  | Const | F4   | As Keys |
| [JScript] | public  | var   | F4   | : Keys; |

*Description*

The F4 key.

*nnn) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F5;     |
| [C++]     | public: | const | Keys | F5;     |
| [VB]      | Public  | Const | F5   | As Keys |
| [JScript] | public  | var   | F5   | : Keys; |

*Description*

The F5 key.

*ooo) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F6;     |
| [C++]     | public: | const | Keys | F6;     |
| [VB]      | Public  | Const | F6   | As Keys |
| [JScript] | public  | var   | F6   | : Keys; |

*Description*

The F6 key.

*ppp) ToString*

[C#]                    public                    const                    Keys                    F7;

[C++]                    public:                    const                    Keys                    F7;

[VB]                    Public                    Const                    F7                    As                    Keys

[JScript]                    public                    var                    F7                    :                    Keys;

*Description*

The F7 key.

*qqq) ToString*

[C#]                    public                    const                    Keys                    F8;

[C++]                    public:                    const                    Keys                    F8;

[VB]                    Public                    Const                    F8                    As                    Keys

[JScript]                    public                    var                    F8                    :                    Keys;

*Description*

The F8 key.

*rrr) ToString*

[C#]                    public                    const                    Keys                    F9;

[C++]                    public:                    const                    Keys                    F9;

|   |           |        |       |    |    |       |
|---|-----------|--------|-------|----|----|-------|
| 1 | [VB]      | Public | Const | F9 | As | Keys  |
| 2 | [JScript] | public | var   | F9 | :  | Keys; |

3

4 *Description*

5 The F9 key.

6 *sss) ToString*

|   |      |        |       |      |            |
|---|------|--------|-------|------|------------|
| 8 | [C#] | public | const | Keys | FinalMode; |
|---|------|--------|-------|------|------------|

|   |       |         |       |      |            |
|---|-------|---------|-------|------|------------|
| 9 | [C++] | public: | const | Keys | FinalMode; |
|---|-------|---------|-------|------|------------|

|    |      |        |       |           |    |      |
|----|------|--------|-------|-----------|----|------|
| 10 | [VB] | Public | Const | FinalMode | As | Keys |
|----|------|--------|-------|-----------|----|------|

|    |           |        |     |           |   |       |
|----|-----------|--------|-----|-----------|---|-------|
| 11 | [JScript] | public | var | FinalMode | : | Keys; |
|----|-----------|--------|-----|-----------|---|-------|

12

13 *Description*

14 The IME final mode key.

15 *ttt) ToString*

|    |      |        |       |      |    |
|----|------|--------|-------|------|----|
| 17 | [C#] | public | const | Keys | G; |
|----|------|--------|-------|------|----|

|    |       |         |       |      |    |
|----|-------|---------|-------|------|----|
| 18 | [C++] | public: | const | Keys | G; |
|----|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 19 | [VB] | Public | Const | G | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 20 | [JScript] | public | var | G | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

21

22 *Description*

23 The G key.

uuu) ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | H;      |
| [C++]     | public: | const | Keys | H;      |
| [VB]      | Public  | Const | H    | As Keys |
| [JScript] | public  | var   | H    | : Keys; |

Description

The H key.

vvv) ToString

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | HanguelMode; |
| [C++]     | public: | const | Keys        | HanguelMode; |
| [VB]      | Public  | Const | HanguelMode | As Keys      |
| [JScript] | public  | var   | HanguelMode | : Keys;      |

Description

The IME Hanguel mode key. (maintained for compatibility; use **HangulMode** )  
The IME Hanguel mode key.

www) ToString

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | HangulMode; |
| [C++]     | public: | const | Keys       | HangulMode; |
| [VB]      | Public  | Const | HangulMode | As Keys     |
| [JScript] | public  | var   | HangulMode | : Keys;     |

*Description*

The IME Hangul mode key.

*xxx) ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | HanjaMode; |
| [C++]     | public: | const | Keys      | HanjaMode; |
| [VB]      | Public  | Const | HanjaMode | As Keys    |
| [JScript] | public  | var   | HanjaMode | : Keys;    |

*Description*

The IME Hanja mode key.

*yyy) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Help;   |
| [C++]     | public: | const | Keys | Help;   |
| [VB]      | Public  | Const | Help | As Keys |
| [JScript] | public  | var   | Help | : Keys; |

*Description*

The HELP key.

*zzz) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Home; |
| [C++] | public: | const | Keys | Home; |

|   |           |        |       |      |    |       |
|---|-----------|--------|-------|------|----|-------|
| 1 | [VB]      | Public | Const | Home | As | Keys  |
| 2 | [JScript] | public | var   | Home | :  | Keys; |

3

4 *Description*

5 The HOME key.

6 **aaaa) ToString**

|    |           |         |       |      |    |       |
|----|-----------|---------|-------|------|----|-------|
| 8  | [C#]      | public  | const | Keys | I; |       |
| 9  | [C++]     | public: | const | Keys | I; |       |
| 10 | [VB]      | Public  | Const | I    | As | Keys  |
| 11 | [JScript] | public  | var   | I    | :  | Keys; |

12

13 *Description*

14 The I key.

15 **bbbb) ToString**

|    |           |         |       |           |            |       |
|----|-----------|---------|-------|-----------|------------|-------|
| 17 | [C#]      | public  | const | Keys      | IMEAccept; |       |
| 18 | [C++]     | public: | const | Keys      | IMEAccept; |       |
| 19 | [VB]      | Public  | Const | IMEAccept | As         | Keys  |
| 20 | [JScript] | public  | var   | IMEAccept | :          | Keys; |

21

22 *Description*

23 The IME Accept key.

*cccc) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | IMEConvert; |
| [C++]     | public: | const | Keys       | IMEConvert; |
| [VB]      | Public  | Const | IMEConvert | As Keys     |
| [JScript] | public  | var   | IMEConvert | : Keys;     |

*Description*

The IME Convert key.

*dddd) ToString*

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C#]      | public  | const | Keys          | IMEModeChange; |
| [C++]     | public: | const | Keys          | IMEModeChange; |
| [VB]      | Public  | Const | IMEModeChange | As Keys        |
| [JScript] | public  | var   | IMEModeChange | : Keys;        |

*Description*

The IME Mode Change key.

*eeee) ToString*

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C#]      | public  | const | Keys          | IMENonconvert; |
| [C++]     | public: | const | Keys          | IMENonconvert; |
| [VB]      | Public  | Const | IMENonconvert | As Keys        |
| [JScript] | public  | var   | IMENonconvert | : Keys;        |

The IME Nonconvert key.

***ffff) ToString***

|      |        |       |      |         |
|------|--------|-------|------|---------|
| [C#] | public | const | Keys | Insert; |
|------|--------|-------|------|---------|

```
[C++]      public:      const      Keys      Insert;
```

|      |        |       |        |    |      |
|------|--------|-------|--------|----|------|
| [VB] | Public | Const | Insert | As | Keys |
|------|--------|-------|--------|----|------|

```
[JScript]      public      var      Insert      :      Keys;
```

### Description

The INS key.

### gggg) ToString

|      |        |       |      |    |
|------|--------|-------|------|----|
| [C#] | public | const | Keys | J; |
|------|--------|-------|------|----|

```
[C++]      public:      const      Keys      J;
```

|      |        |       |   |    |      |
|------|--------|-------|---|----|------|
| [VB] | Public | Const | J | As | Keys |
|------|--------|-------|---|----|------|

```
[JScript]      public      var      J      :      Keys;
```

*Description*

The J key.

### *hhhh) ToString*

|      |        |       |      |            |
|------|--------|-------|------|------------|
| [C#] | public | const | Keys | JunjaMode; |
|------|--------|-------|------|------------|

```
[C++]      public:      const      Keys      JunjaMode;
```



|           |        |       |           |    |       |
|-----------|--------|-------|-----------|----|-------|
| [VB]      | Public | Const | JunjaMode | As | Keys  |
| [JScript] | public | var   | JunjaMode | :  | Keys; |

*Description*

The IME Junja mode key.

*iii) ToString*

|           |         |       |      |    |       |
|-----------|---------|-------|------|----|-------|
| [C#]      | public  | const | Keys | K; |       |
| [C++]     | public: | const | Keys | K; |       |
| [VB]      | Public  | Const | K    | As | Keys  |
| [JScript] | public  | var   | K    | :  | Keys; |

*Description*

The K key.

*jjjj) ToString*

|           |         |       |          |           |       |
|-----------|---------|-------|----------|-----------|-------|
| [C#]      | public  | const | Keys     | KanaMode; |       |
| [C++]     | public: | const | Keys     | KanaMode; |       |
| [VB]      | Public  | Const | KanaMode | As        | Keys  |
| [JScript] | public  | var   | KanaMode | :         | Keys; |

*Description*

The IME Kana mode key.

**kkkk) ToString**

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | KanjiMode; |
| [C++]     | public: | const | Keys      | KanjiMode; |
| [VB]      | Public  | Const | KanjiMode | As Keys    |
| [JScript] | public  | var   | KanjiMode | : Keys;    |

**Description**

The IME Kanji mode key.

**llll) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | KeyCode; |
| [C++]     | public: | const | Keys    | KeyCode; |
| [VB]      | Public  | Const | KeyCode | As Keys  |
| [JScript] | public  | var   | KeyCode | : Keys;  |

**Description**

The bitmask to extract a key code from a key value.

**mmmm)ToString**

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | L;      |
| [C++]     | public: | const | Keys | L;      |
| [VB]      | Public  | Const | L    | As Keys |
| [JScript] | public  | var   | L    | : Keys; |

*Description*

The L key.

*nnnn) ToString*

[C#]            public            const            Keys            LaunchApplication1;

[C++]           public:            const            Keys            LaunchApplication1;

[VB]           Public           Const           LaunchApplication1           As           Keys

[JScript]       public            var            LaunchApplication1           :           Keys;

*Description*

The Start Application one key (Windows 2000 or later).

*oooo) ToString*

[C#]            public            const            Keys            LaunchApplication2;

[C++]           public:            const            Keys            LaunchApplication2;

[VB]           Public           Const           LaunchApplication2           As           Keys

[JScript]       public            var            LaunchApplication2           :           Keys;

*Description*

The Start Application two key (Windows 2000 or later).

*pppp) ToString*

[C#]            public            const            Keys            LaunchMail;

[C++]           public:            const            Keys            LaunchMail;

|           |        |       |            |    |       |
|-----------|--------|-------|------------|----|-------|
| [VB]      | Public | Const | LaunchMail | As | Keys  |
| [JScript] | public | var   | LaunchMail | :  | Keys; |

*Description*

The Launch Mail key (Windows 2000 or later).

*qqqq) ToString*

|           |         |       |         |          |       |
|-----------|---------|-------|---------|----------|-------|
| [C#]      | public  | const | Keys    | LButton; |       |
| [C++]     | public: | const | Keys    | LButton; |       |
| [VB]      | Public  | Const | LButton | As       | Keys  |
| [JScript] | public  | var   | LButton | :        | Keys; |

*Description*

The left mouse button.

*rrrr) ToString*

|           |         |       |             |              |       |
|-----------|---------|-------|-------------|--------------|-------|
| [C#]      | public  | const | Keys        | LControlKey; |       |
| [C++]     | public: | const | Keys        | LControlKey; |       |
| [VB]      | Public  | Const | LControlKey | As           | Keys  |
| [JScript] | public  | var   | LControlKey | :            | Keys; |

*Description*

The left CTRL key.

**ssss) ToString**

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Left;   |
| [C++]     | public: | const | Keys | Left;   |
| [VB]      | Public  | Const | Left | As Keys |
| [JScript] | public  | var   | Left | : Keys; |

**Description**

The LEFT ARROW key.

**tttt) ToString**

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | LineFeed; |
| [C++]     | public: | const | Keys     | LineFeed; |
| [VB]      | Public  | Const | LineFeed | As Keys   |
| [JScript] | public  | var   | LineFeed | : Keys;   |

**Description**

The LINEFEED key.

**uuuu) ToString**

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | LMenu;  |
| [C++]     | public: | const | Keys  | LMenu;  |
| [VB]      | Public  | Const | LMenu | As Keys |
| [JScript] | public  | var   | LMenu | : Keys; |

1  
2 *Description*

3 The left ALT key.

4 *vvvv) ToString*

5  
6 [C#] public const Keys LShiftKey;

7 [C++] public: const Keys LShiftKey;

8 [VB] Public Const LShiftKey As Keys

9 [JScript] public var LShiftKey : Keys;

10  
11 *Description*

12 The left SHIFT key.

13 *www)ToString*

14  
15 [C#] public const Keys LWin;

16 [C++] public: const Keys LWin;

17 [VB] Public Const LWin As Keys

18 [JScript] public var LWin : Keys;

19  
20 *Description*

21 The left Windows logo key (Microsoft Natural Keyboard).

22 *xxxx) ToString*

23  
24 [C#] public const Keys M;

25 [C++] public: const Keys M;

|    |                                                   |         |                 |                |    |                 |
|----|---------------------------------------------------|---------|-----------------|----------------|----|-----------------|
| 1  | [VB]                                              | Public  | Const           | M              | As | Keys            |
| 2  | [JScript]                                         | public  | var             | M              | :  | Keys;           |
| 3  |                                                   |         |                 |                |    |                 |
| 4  | <i>Description</i>                                |         |                 |                |    |                 |
| 5  | The M key.                                        |         |                 |                |    |                 |
| 6  |                                                   | yyyy)   | <i>ToString</i> |                |    |                 |
| 7  |                                                   |         |                 |                |    |                 |
| 8  | [C#]                                              | public  | const           | Keys           |    | MButton;        |
| 9  | [C++]                                             | public: | const           | Keys           |    | MButton;        |
| 10 | [VB]                                              | Public  | Const           | MButton        | As | Keys            |
| 11 | [JScript]                                         | public  | var             | MButton        | :  | Keys;           |
| 12 |                                                   |         |                 |                |    |                 |
| 13 | <i>Description</i>                                |         |                 |                |    |                 |
| 14 | The middle mouse button (three-button mouse).     |         |                 |                |    |                 |
| 15 |                                                   | zzzz)   | <i>ToString</i> |                |    |                 |
| 16 |                                                   |         |                 |                |    |                 |
| 17 | [C#]                                              | public  | const           | Keys           |    | MediaNextTrack; |
| 18 | [C++]                                             | public: | const           | Keys           |    | MediaNextTrack; |
| 19 | [VB]                                              | Public  | Const           | MediaNextTrack | As | Keys            |
| 20 | [JScript]                                         | public  | var             | MediaNextTrack | :  | Keys;           |
| 21 |                                                   |         |                 |                |    |                 |
| 22 | <i>Description</i>                                |         |                 |                |    |                 |
| 23 | The Media Next Track key (Windows 2000 or later). |         |                 |                |    |                 |
| 24 |                                                   |         |                 |                |    |                 |
| 25 |                                                   |         |                 |                |    |                 |

***aaaaa) ToString***

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | Keys           | MediaPlayPause; |
| [C++]     | public: | const | Keys           | MediaPlayPause; |
| [VB]      | Public  | Const | MediaPlayPause | As Keys         |
| [JScript] | public  | var   | MediaPlayPause | : Keys;         |

***Description***

The Media Play Pause key (Windows 2000 or later).

***bbbbbb) ToString***

|           |         |       |                    |                     |
|-----------|---------|-------|--------------------|---------------------|
| [C#]      | public  | const | Keys               | MediaPreviousTrack; |
| [C++]     | public: | const | Keys               | MediaPreviousTrack; |
| [VB]      | Public  | Const | MediaPreviousTrack | As Keys             |
| [JScript] | public  | var   | MediaPreviousTrack | : Keys;             |

***Description***

The Media Previous Track key (Windows 2000 or later).

***cccccc) ToString***

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | MediaStop; |
| [C++]     | public: | const | Keys      | MediaStop; |
| [VB]      | Public  | Const | MediaStop | As Keys    |
| [JScript] | public  | var   | MediaStop | : Keys;    |



*Description*

The Media Stop key (Windows 2000 or later).

*dddd) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Menu;   |
| [C++]     | public: | const | Keys | Menu;   |
| [VB]      | Public  | Const | Menu | As Keys |
| [JScript] | public  | var   | Menu | : Keys; |

*Description*

The ALT key.

*eeee) ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | Modifiers; |
| [C++]     | public: | const | Keys      | Modifiers; |
| [VB]      | Public  | Const | Modifiers | As Keys    |
| [JScript] | public  | var   | Modifiers | : Keys;    |

*Description*

The bitmask to extract modifiers from a key value.

*ffff) ToString*

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C#]  | public  | const | Keys | Multiply; |
| [C++] | public: | const | Keys | Multiply; |

|    |                        |         |       |          |    |       |
|----|------------------------|---------|-------|----------|----|-------|
| 1  | [VB]                   | Public  | Const | Multiply | As | Keys  |
| 2  | [JScript]              | public  | var   | Multiply | :  | Keys; |
| 3  |                        |         |       |          |    |       |
| 4  | <i>Description</i>     |         |       |          |    |       |
| 5  | The Multiply key.      |         |       |          |    |       |
| 6  | <b>ggggg) ToString</b> |         |       |          |    |       |
| 7  |                        |         |       |          |    |       |
| 8  | [C#]                   | public  | const | Keys     |    | N;    |
| 9  | [C++]                  | public: | const | Keys     |    | N;    |
| 10 | [VB]                   | Public  | Const | N        | As | Keys  |
| 11 | [JScript]              | public  | var   | N        | :  | Keys; |
| 12 |                        |         |       |          |    |       |
| 13 | <i>Description</i>     |         |       |          |    |       |
| 14 | The N key.             |         |       |          |    |       |
| 15 | <b>hhhhh) ToString</b> |         |       |          |    |       |
| 16 |                        |         |       |          |    |       |
| 17 | [C#]                   | public  | const | Keys     |    | Next; |
| 18 | [C++]                  | public: | const | Keys     |    | Next; |
| 19 | [VB]                   | Public  | Const | Next     | As | Keys  |
| 20 | [JScript]              | public  | var   | Next     | :  | Keys; |
| 21 |                        |         |       |          |    |       |
| 22 | <i>Description</i>     |         |       |          |    |       |
| 23 | The PAGE DOWN key.     |         |       |          |    |       |
| 24 |                        |         |       |          |    |       |
| 25 |                        |         |       |          |    |       |

iiii) ToString

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | NoName; |
| [C++]     | public: | const | Keys   | NoName; |
| [VB]      | Public  | Const | NoName | As Keys |
| [JScript] | public  | var   | NoName | : Keys; |

Description

A constant reserved for future use.

jjjj) ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | None;   |
| [C++]     | public: | const | Keys | None;   |
| [VB]      | Public  | Const | None | As Keys |
| [JScript] | public  | var   | None | : Keys; |

Description

No key pressed.

kkkkk) ToString

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumLock; |
| [C++]     | public: | const | Keys    | NumLock; |
| [VB]      | Public  | Const | NumLock | As Keys  |
| [JScript] | public  | var   | NumLock | : Keys;  |

*Description*

The NUM LOCK key.

*llll) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad0; |
| [C++]     | public: | const | Keys    | NumPad0; |
| [VB]      | Public  | Const | NumPad0 | As Keys  |
| [JScript] | public  | var   | NumPad0 | : Keys;  |

*Description*

The 0 key on the numeric keypad.

*mmmmm)ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad1; |
| [C++]     | public: | const | Keys    | NumPad1; |
| [VB]      | Public  | Const | NumPad1 | As Keys  |
| [JScript] | public  | var   | NumPad1 | : Keys;  |

*Description*

The 1 key on the numeric keypad.

*nnnnn)ToString*

|       |         |       |      |          |
|-------|---------|-------|------|----------|
| [C#]  | public  | const | Keys | NumPad2; |
| [C++] | public: | const | Keys | NumPad2; |

|           |        |       |         |    |       |
|-----------|--------|-------|---------|----|-------|
| [VB]      | Public | Const | NumPad2 | As | Keys  |
| [JScript] | public | var   | NumPad2 | :  | Keys; |

*Description*

The 2 key on the numeric keypad.

**ooooo) ToString**

|           |         |       |         |    |          |
|-----------|---------|-------|---------|----|----------|
| [C#]      | public  | const | Keys    |    | NumPad3; |
| [C++]     | public: | const | Keys    |    | NumPad3; |
| [VB]      | Public  | Const | NumPad3 | As | Keys     |
| [JScript] | public  | var   | NumPad3 | :  | Keys;    |

*Description*

The 3 key on the numeric keypad.

**ppppp) ToString**

|           |         |       |         |    |          |
|-----------|---------|-------|---------|----|----------|
| [C#]      | public  | const | Keys    |    | NumPad4; |
| [C++]     | public: | const | Keys    |    | NumPad4; |
| [VB]      | Public  | Const | NumPad4 | As | Keys     |
| [JScript] | public  | var   | NumPad4 | :  | Keys;    |

*Description*

The 4 key on the numeric keypad.

*qqqqq) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad5; |
| [C++]     | public: | const | Keys    | NumPad5; |
| [VB]      | Public  | Const | NumPad5 | As Keys  |
| [JScript] | public  | var   | NumPad5 | : Keys;  |

*Description*

The 5 key on the numeric keypad.

*rrrrr) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad6; |
| [C++]     | public: | const | Keys    | NumPad6; |
| [VB]      | Public  | Const | NumPad6 | As Keys  |
| [JScript] | public  | var   | NumPad6 | : Keys;  |

*Description*

The 6 key on the numeric keypad.

*sssss) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad7; |
| [C++]     | public: | const | Keys    | NumPad7; |
| [VB]      | Public  | Const | NumPad7 | As Keys  |
| [JScript] | public  | var   | NumPad7 | : Keys;  |

*Description*

The 7 key on the numeric keypad.

*ttttt) ToString*

[C#]            public            const            Keys            NumPad8;

[C++]           public:            const            Keys            NumPad8;

[VB]            Public            Const            NumPad8           As            Keys

[JScript]       public            var            NumPad8           :            Keys;

*Description*

The 8 key on the numeric keypad.

*uuuuu)ToString*

[C#]            public            const            Keys            NumPad9;

[C++]           public:            const            Keys            NumPad9;

[VB]            Public            Const            NumPad9           As            Keys

[JScript]       public            var            NumPad9           :            Keys;

*Description*

The 9 key on the numeric keypad.

*vvvvv) ToString*

[C#]            public            const            Keys            O;

[C++]           public:            const            Keys            O;

|    |                                                                                            |         |       |              |    |               |
|----|--------------------------------------------------------------------------------------------|---------|-------|--------------|----|---------------|
| 1  | [VB]                                                                                       | Public  | Const | O            | As | Keys          |
| 2  | [JScript]                                                                                  | public  | var   | O            | :  | Keys;         |
| 3  |                                                                                            |         |       |              |    |               |
| 4  | <i>Description</i>                                                                         |         |       |              |    |               |
| 5  | The O key.                                                                                 |         |       |              |    |               |
| 6  | <i>wwwwww)ToString</i>                                                                     |         |       |              |    |               |
| 7  |                                                                                            |         |       |              |    |               |
| 8  | [C#]                                                                                       | public  | const | Keys         |    | Oem8;         |
| 9  | [C++]                                                                                      | public: | const | Keys         |    | Oem8;         |
| 10 | [VB]                                                                                       | Public  | Const | Oem8         | As | Keys          |
| 11 | [JScript]                                                                                  | public  | var   | Oem8         | :  | Keys;         |
| 12 |                                                                                            |         |       |              |    |               |
| 13 | <i>Description</i>                                                                         |         |       |              |    |               |
| 14 | OEM specific.                                                                              |         |       |              |    |               |
| 15 | <i>xxxxx) ToString</i>                                                                     |         |       |              |    |               |
| 16 |                                                                                            |         |       |              |    |               |
| 17 | [C#]                                                                                       | public  | const | Keys         |    | OemBackslash; |
| 18 | [C++]                                                                                      | public: | const | Keys         |    | OemBackslash; |
| 19 | [VB]                                                                                       | Public  | Const | OemBackslash | As | Keys          |
| 20 | [JScript]                                                                                  | public  | var   | OemBackslash | :  | Keys;         |
| 21 |                                                                                            |         |       |              |    |               |
| 22 | <i>Description</i>                                                                         |         |       |              |    |               |
| 23 | The OEM Angle bracket or Backslash key on the RT 102 key keyboard (Windows 2000 or later). |         |       |              |    |               |
| 24 |                                                                                            |         |       |              |    |               |
| 25 |                                                                                            |         |       |              |    |               |



yyyyy) ToString

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | OemClear; |
| [C++]     | public: | const | Keys     | OemClear; |
| [VB]      | Public  | Const | OemClear | As Keys   |
| [JScript] | public  | var   | OemClear | : Keys;   |

Description

The CLEAR key.

zzzzz) ToString

|           |         |       |                  |                   |
|-----------|---------|-------|------------------|-------------------|
| [C#]      | public  | const | Keys             | OemCloseBrackets; |
| [C++]     | public: | const | Keys             | OemCloseBrackets; |
| [VB]      | Public  | Const | OemCloseBrackets | As Keys           |
| [JScript] | public  | var   | OemCloseBrackets | : Keys;           |

Description

The OEM Close Bracket key on a US standard keyboard (Windows 2000 or later).

aaaaaa)ToString

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | Oemcomma; |
| [C++]     | public: | const | Keys     | Oemcomma; |
| [VB]      | Public  | Const | Oemcomma | As Keys   |
| [JScript] | public  | var   | Oemcomma | : Keys;   |

*Description*

The OEM Comma key on any country/region keyboard (Windows 2000 or later).

*bbbbbb)ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | OemMinus; |
| [C++]     | public: | const | Keys     | OemMinus; |
| [VB]      | Public  | Const | OemMinus | As Keys   |
| [JScript] | public  | var   | OemMinus | : Keys;   |

*Description*

The OEM Minus key on any country/region keyboard (Windows 2000 or later).

*cccccc)ToString*

|           |         |       |                 |                  |
|-----------|---------|-------|-----------------|------------------|
| [C#]      | public  | const | Keys            | OemOpenBrackets; |
| [C++]     | public: | const | Keys            | OemOpenBrackets; |
| [VB]      | Public  | Const | OemOpenBrackets | As Keys          |
| [JScript] | public  | var   | OemOpenBrackets | : Keys;          |

*Description*

The OEM OpenBracket key on a US standard keyboard (Windows 2000 or later).

*dddddd)ToString*

|       |         |       |      |            |
|-------|---------|-------|------|------------|
| [C#]  | public  | const | Keys | OemPeriod; |
| [C++] | public: | const | Keys | OemPeriod; |

|           |        |       |           |    |       |
|-----------|--------|-------|-----------|----|-------|
| [VB]      | Public | Const | OemPeriod | As | Keys  |
| [JScript] | public | var   | OemPeriod | :  | Keys; |

*Description*

The OEM Period key on any country/region keyboard (Windows 2000 or later).

*eeeeee)ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | OemPipe; |
| [C++]     | public: | const | Keys    | OemPipe; |
| [VB]      | Public  | Const | OemPipe | As Keys  |
| [JScript] | public  | var   | OemPipe | : Keys;  |

*Description*

The OEM Pipe key on a US standard keyboard (Windows 2000 or later).

*ffffff) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Oemplus; |
| [C++]     | public: | const | Keys    | Oemplus; |
| [VB]      | Public  | Const | Oemplus | As Keys  |
| [JScript] | public  | var   | Oemplus | : Keys;  |

*Description*

The OEM Plus key on any country/region keyboard (Windows 2000 or later).

**gggggg)ToString**

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | OemQuestion; |
| [C++]     | public: | const | Keys        | OemQuestion; |
| [VB]      | Public  | Const | OemQuestion | As Keys      |
| [JScript] | public  | var   | OemQuestion | : Keys;      |

**Description**

The OEM Question Mark key on a US standard keyboard (Windows 2000 or later).

**hhhhh)ToString**

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | OemQuotes; |
| [C++]     | public: | const | Keys      | OemQuotes; |
| [VB]      | Public  | Const | OemQuotes | As Keys    |
| [JScript] | public  | var   | OemQuotes | : Keys;    |

**Description**

The OEM Singled/Double quote key on a US standard keyboard (Windows 2000 or later).

**iiiiii) ToString**

|           |         |       |              |               |
|-----------|---------|-------|--------------|---------------|
| [C#]      | public  | const | Keys         | OemSemicolon; |
| [C++]     | public: | const | Keys         | OemSemicolon; |
| [VB]      | Public  | Const | OemSemicolon | As Keys       |
| [JScript] | public  | var   | OemSemicolon | : Keys;       |

*Description*

The OEM Semicolon key on a US standard keyboard (Windows 2000 or later).

*jjjjj) ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | Oemtilde; |
| [C++]     | public: | const | Keys     | Oemtilde; |
| [VB]      | Public  | Const | Oemtilde | As Keys   |
| [JScript] | public  | var   | Oemtilde | : Keys;   |

*Description*

The OEM tilde key on a US standard keyboard (Windows 2000 or later).

*kkkkkk)ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | P;      |
| [C++]     | public: | const | Keys | P;      |
| [VB]      | Public  | Const | P    | As Keys |
| [JScript] | public  | var   | P    | : Keys; |

*Description*

The P key.

*lllll) ToString*

|       |         |       |      |      |
|-------|---------|-------|------|------|
| [C#]  | public  | const | Keys | Pal; |
| [C++] | public: | const | Keys | Pal; |

|    |                        |         |       |          |    |           |
|----|------------------------|---------|-------|----------|----|-----------|
| 1  | [VB]                   | Public  | Const | Pa1      | As | Keys      |
| 2  | [JScript]              | public  | var   | Pa1      | :  | Keys;     |
| 3  |                        |         |       |          |    |           |
| 4  | <i>Description</i>     |         |       |          |    |           |
| 5  | The PA1 key.           |         |       |          |    |           |
| 6  | <i>mmmmmm)ToString</i> |         |       |          |    |           |
| 7  |                        |         |       |          |    |           |
| 8  | [C#]                   | public  | const | Keys     |    | PageDown; |
| 9  | [C++]                  | public: | const | Keys     |    | PageDown; |
| 10 | [VB]                   | Public  | Const | PageDown | As | Keys      |
| 11 | [JScript]              | public  | var   | PageDown | :  | Keys;     |
| 12 |                        |         |       |          |    |           |
| 13 | <i>Description</i>     |         |       |          |    |           |
| 14 | The PAGE DOWN key.     |         |       |          |    |           |
| 15 | <i>nnnnnn)ToString</i> |         |       |          |    |           |
| 16 |                        |         |       |          |    |           |
| 17 | [C#]                   | public  | const | Keys     |    | PageUp;   |
| 18 | [C++]                  | public: | const | Keys     |    | PageUp;   |
| 19 | [VB]                   | Public  | Const | PageUp   | As | Keys      |
| 20 | [JScript]              | public  | var   | PageUp   | :  | Keys;     |
| 21 |                        |         |       |          |    |           |
| 22 | <i>Description</i>     |         |       |          |    |           |
| 23 | The PAGE UP key.       |         |       |          |    |           |
| 24 |                        |         |       |          |    |           |
| 25 |                        |         |       |          |    |           |

*oooooo)ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Pause;  |
| [C++]     | public: | const | Keys  | Pause;  |
| [VB]      | Public  | Const | Pause | As Keys |
| [JScript] | public  | var   | Pause | : Keys; |

*Description*

The PAUSE key.

*pppppp)ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Play;   |
| [C++]     | public: | const | Keys | Play;   |
| [VB]      | Public  | Const | Play | As Keys |
| [JScript] | public  | var   | Play | : Keys; |

*Description*

The PLAY key.

*qqqqqq)ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Print;  |
| [C++]     | public: | const | Keys  | Print;  |
| [VB]      | Public  | Const | Print | As Keys |
| [JScript] | public  | var   | Print | : Keys; |

1  
2 *Description*

3 The PRINT key.

4 *rrrrrr) ToString*

5  
6 [C#] public const Keys PrintScreen;

7 [C++] public: const Keys PrintScreen;

8 [VB] Public Const PrintScreen As Keys

9 [JScript] public var PrintScreen : Keys;

10  
11 *Description*

12 The PRINT SCREEN key.

13 *ssssss) ToString*

14  
15 [C#] public const Keys Prior;

16 [C++] public: const Keys Prior;

17 [VB] Public Const Prior As Keys

18 [JScript] public var Prior : Keys;

19  
20 *Description*

21 The PAGE UP key.

22 *tttttt) ToString*

23  
24 [C#] public const Keys ProcessKey;

25 [C++] public: const Keys ProcessKey;



|   |           |        |       |            |    |       |
|---|-----------|--------|-------|------------|----|-------|
| 1 | [VB]      | Public | Const | ProcessKey | As | Keys  |
| 2 | [JScript] | public | var   | ProcessKey | :  | Keys; |

3

*Description*

The PROCESS KEY key.

*uuuuuu)ToString*

7

|   |      |        |       |      |    |
|---|------|--------|-------|------|----|
| 8 | [C#] | public | const | Keys | Q; |
|---|------|--------|-------|------|----|

|   |       |         |       |      |    |
|---|-------|---------|-------|------|----|
| 9 | [C++] | public: | const | Keys | Q; |
|---|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 10 | [VB] | Public | Const | Q | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 11 | [JScript] | public | var | Q | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

12

*Description*

The Q key.

*vvvvvv)ToString*

16

|    |      |        |       |      |    |
|----|------|--------|-------|------|----|
| 17 | [C#] | public | const | Keys | R; |
|----|------|--------|-------|------|----|

|    |       |         |       |      |    |
|----|-------|---------|-------|------|----|
| 18 | [C++] | public: | const | Keys | R; |
|----|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 19 | [VB] | Public | Const | R | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 20 | [JScript] | public | var | R | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

21

*Description*

The R key.

24

25

*wwwww)ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | RButton; |
| [C++]     | public: | const | Keys    | RButton; |
| [VB]      | Public  | Const | RButton | As Keys  |
| [JScript] | public  | var   | RButton | : Keys;  |

*Description*

The right mouse button.

*xxxxxx)ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | RControlKey; |
| [C++]     | public: | const | Keys        | RControlKey; |
| [VB]      | Public  | Const | RControlKey | As Keys      |
| [JScript] | public  | var   | RControlKey | : Keys;      |

*Description*

The right CTRL key.

*yyyyyy)ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Return; |
| [C++]     | public: | const | Keys   | Return; |
| [VB]      | Public  | Const | Return | As Keys |
| [JScript] | public  | var   | Return | : Keys; |

*Description*

The RETURN key.

*zzzzzz) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Right;  |
| [C++]     | public: | const | Keys  | Right;  |
| [VB]      | Public  | Const | Right | As Keys |
| [JScript] | public  | var   | Right | : Keys; |

*Description*

The RIGHT ARROW key.

*aaaaaaa) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | RMenu;  |
| [C++]     | public: | const | Keys  | RMenu;  |
| [VB]      | Public  | Const | RMenu | As Keys |
| [JScript] | public  | var   | RMenu | : Keys; |

*Description*

The right ALT key.

*bbbbbbb) ToString*

|       |         |       |      |            |
|-------|---------|-------|------|------------|
| [C#]  | public  | const | Keys | RShiftKey; |
| [C++] | public: | const | Keys | RShiftKey; |

|   |           |        |       |           |    |       |
|---|-----------|--------|-------|-----------|----|-------|
| 1 | [VB]      | Public | Const | RShiftKey | As | Keys  |
| 2 | [JScript] | public | var   | RShiftKey | :  | Keys; |

3

4 *Description*

5 The right SHIFT key.

6 *ccccccc)ToString*

7

|   |      |        |       |      |       |
|---|------|--------|-------|------|-------|
| 8 | [C#] | public | const | Keys | RWin; |
|---|------|--------|-------|------|-------|

|   |       |         |       |      |       |
|---|-------|---------|-------|------|-------|
| 9 | [C++] | public: | const | Keys | RWin; |
|---|-------|---------|-------|------|-------|

|    |      |        |       |      |    |      |
|----|------|--------|-------|------|----|------|
| 10 | [VB] | Public | Const | RWin | As | Keys |
|----|------|--------|-------|------|----|------|

|    |           |        |     |      |   |       |
|----|-----------|--------|-----|------|---|-------|
| 11 | [JScript] | public | var | RWin | : | Keys; |
|----|-----------|--------|-----|------|---|-------|

12

13 *Description*

14 The right Windows logo key (Microsoft Natural Keyboard).

15 *ddddddd)ToString*

16

|    |      |        |       |      |    |
|----|------|--------|-------|------|----|
| 17 | [C#] | public | const | Keys | S; |
|----|------|--------|-------|------|----|

|    |       |         |       |      |    |
|----|-------|---------|-------|------|----|
| 18 | [C++] | public: | const | Keys | S; |
|----|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 19 | [VB] | Public | Const | S | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 20 | [JScript] | public | var | S | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

21

22 *Description*

23 The S key.

24

25

*eeeeeee)ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Scroll; |
| [C++]     | public: | const | Keys   | Scroll; |
| [VB]      | Public  | Const | Scroll | As Keys |
| [JScript] | public  | var   | Scroll | : Keys; |

*Description*

The SCROLL LOCK key.

*ffffff) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Select; |
| [C++]     | public: | const | Keys   | Select; |
| [VB]      | Public  | Const | Select | As Keys |
| [JScript] | public  | var   | Select | : Keys; |

*Description*

The SELECT key.

*ggggggg)ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | SelectMedia; |
| [C++]     | public: | const | Keys        | SelectMedia; |
| [VB]      | Public  | Const | SelectMedia | As Keys      |
| [JScript] | public  | var   | SelectMedia | : Keys;      |

*Description*

The Select Media key (Windows 2000 or later).

*hhhhhh)ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | Separator; |
| [C++]     | public: | const | Keys      | Separator; |
| [VB]      | Public  | Const | Separator | As Keys    |
| [JScript] | public  | var   | Separator | : Keys;    |

*Description*

The Separator key.

*iiiiii) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Shift;  |
| [C++]     | public: | const | Keys  | Shift;  |
| [VB]      | Public  | Const | Shift | As Keys |
| [JScript] | public  | var   | Shift | : Keys; |

*Description*

The SHIFT modifier key.

*jjjjjj) ToString*

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C#]  | public  | const | Keys | ShiftKey; |
| [C++] | public: | const | Keys | ShiftKey; |

|    |                          |         |       |          |    |           |
|----|--------------------------|---------|-------|----------|----|-----------|
| 1  | [VB]                     | Public  | Const | ShiftKey | As | Keys      |
| 2  | [JScript]                | public  | var   | ShiftKey | :  | Keys;     |
| 3  |                          |         |       |          |    |           |
| 4  | <i>Description</i>       |         |       |          |    |           |
| 5  | The SHIFT key.           |         |       |          |    |           |
| 6  | <i>kkkkkkk)ToString</i>  |         |       |          |    |           |
| 7  |                          |         |       |          |    |           |
| 8  | [C#]                     | public  | const | Keys     |    | Snapshot; |
| 9  | [C++]                    | public: | const | Keys     |    | Snapshot; |
| 10 | [VB]                     | Public  | Const | Snapshot | As | Keys      |
| 11 | [JScript]                | public  | var   | Snapshot | :  | Keys;     |
| 12 |                          |         |       |          |    |           |
| 13 | <i>Description</i>       |         |       |          |    |           |
| 14 | The PRINT SCREEN key.    |         |       |          |    |           |
| 15 | <i>lllllll) ToString</i> |         |       |          |    |           |
| 16 |                          |         |       |          |    |           |
| 17 | [C#]                     | public  | const | Keys     |    | Space;    |
| 18 | [C++]                    | public: | const | Keys     |    | Space;    |
| 19 | [VB]                     | Public  | Const | Space    | As | Keys      |
| 20 | [JScript]                | public  | var   | Space    | :  | Keys;     |
| 21 |                          |         |       |          |    |           |
| 22 | <i>Description</i>       |         |       |          |    |           |
| 23 | The SPACEBAR key.        |         |       |          |    |           |
| 24 |                          |         |       |          |    |           |
| 25 |                          |         |       |          |    |           |

***mmmmmm)ToString***

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | Subtract; |
| [C++]     | public: | const | Keys     | Subtract; |
| [VB]      | Public  | Const | Subtract | As Keys   |
| [JScript] | public  | var   | Subtract | : Keys;   |

***Description***

The Subtract key.

***nnnnnnn)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | T;      |
| [C++]     | public: | const | Keys | T;      |
| [VB]      | Public  | Const | T    | As Keys |
| [JScript] | public  | var   | T    | : Keys; |

***Description***

The T key.

***ooooooo)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Tab;    |
| [C++]     | public: | const | Keys | Tab;    |
| [VB]      | Public  | Const | Tab  | As Keys |
| [JScript] | public  | var   | Tab  | : Keys; |



*Description*

The TAB key.

***ppppppp)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | U;      |
| [C++]     | public: | const | Keys | U;      |
| [VB]      | Public  | Const | U    | As Keys |
| [JScript] | public  | var   | U    | : Keys; |

*Description*

The U key.

***qqqqqqq)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Up;     |
| [C++]     | public: | const | Keys | Up;     |
| [VB]      | Public  | Const | Up   | As Keys |
| [JScript] | public  | var   | Up   | : Keys; |

*Description*

The UP ARROW key.

***rrrrrrr)ToString***

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C#]  | public  | const | Keys | V; |
| [C++] | public: | const | Keys | V; |

|           |        |       |   |    |       |
|-----------|--------|-------|---|----|-------|
| [VB]      | Public | Const | V | As | Keys  |
| [JScript] | public | var   | V | :  | Keys; |

*Description*

The V key.

*sssssss)ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | VolumeDown; |
| [C++]     | public: | const | Keys       | VolumeDown; |
| [VB]      | Public  | Const | VolumeDown | As Keys     |
| [JScript] | public  | var   | VolumeDown | : Keys;     |

*Description*

The Volume Down key (Windows 2000 or later).

*tttttt) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | VolumeMute; |
| [C++]     | public: | const | Keys       | VolumeMute; |
| [VB]      | Public  | Const | VolumeMute | As Keys     |
| [JScript] | public  | var   | VolumeMute | : Keys;     |

*Description*

The Volume Mute key (Windows 2000 or later).

uuuuuuu)ToString

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | VolumeUp; |
| [C++]     | public: | const | Keys     | VolumeUp; |
| [VB]      | Public  | Const | VolumeUp | As Keys   |
| [JScript] | public  | var   | VolumeUp | : Keys;   |

Description

The Volume Up key (Windows 2000 or later).

vvvvvvv)ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | W;      |
| [C++]     | public: | const | Keys | W;      |
| [VB]      | Public  | Const | W    | As Keys |
| [JScript] | public  | var   | W    | : Keys; |

Description

The W key.

wwwwwww)ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | X;      |
| [C++]     | public: | const | Keys | X;      |
| [VB]      | Public  | Const | X    | As Keys |
| [JScript] | public  | var   | X    | : Keys; |

*Description*

The X key.

*xxxxxxx)ToString*

|      |        |       |      |           |
|------|--------|-------|------|-----------|
| [C#] | public | const | Keys | XButton1; |
|------|--------|-------|------|-----------|

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C++] | public: | const | Keys | XButton1; |
|-------|---------|-------|------|-----------|

|      |        |       |          |    |      |
|------|--------|-------|----------|----|------|
| [VB] | Public | Const | XButton1 | As | Keys |
|------|--------|-------|----------|----|------|

|           |        |     |          |   |       |
|-----------|--------|-----|----------|---|-------|
| [JScript] | public | var | XButton1 | : | Keys; |
|-----------|--------|-----|----------|---|-------|

*Description*

The first x mouse button (five-button mouse).

*yyyyyyy)ToString*

|      |        |       |      |           |
|------|--------|-------|------|-----------|
| [C#] | public | const | Keys | XButton2; |
|------|--------|-------|------|-----------|

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C++] | public: | const | Keys | XButton2; |
|-------|---------|-------|------|-----------|

|      |        |       |          |    |      |
|------|--------|-------|----------|----|------|
| [VB] | Public | Const | XButton2 | As | Keys |
|------|--------|-------|----------|----|------|

|           |        |     |          |   |       |
|-----------|--------|-----|----------|---|-------|
| [JScript] | public | var | XButton2 | : | Keys; |
|-----------|--------|-----|----------|---|-------|

*Description*

The second x mouse button (five-button mouse).

*zzzzzzz)ToString*

|      |        |       |      |    |
|------|--------|-------|------|----|
| [C#] | public | const | Keys | Y; |
|------|--------|-------|------|----|

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C++] | public: | const | Keys | Y; |
|-------|---------|-------|------|----|

|    |                           |         |       |      |      |       |
|----|---------------------------|---------|-------|------|------|-------|
| 1  | [VB]                      | Public  | Const | Y    | As   | Keys  |
| 2  | [JScript]                 | public  | var   | Y    | :    | Keys; |
| 3  |                           |         |       |      |      |       |
| 4  | <i>Description</i>        |         |       |      |      |       |
| 5  | The Y key.                |         |       |      |      |       |
| 6  | <b>aaaaaaaaa)ToString</b> |         |       |      |      |       |
| 7  |                           |         |       |      |      |       |
| 8  | [C#]                      | public  | const |      | Keys | Z;    |
| 9  | [C++]                     | public: | const |      | Keys | Z;    |
| 10 | [VB]                      | Public  | Const | Z    | As   | Keys  |
| 11 | [JScript]                 | public  | var   | Z    | :    | Keys; |
| 12 |                           |         |       |      |      |       |
| 13 | <i>Description</i>        |         |       |      |      |       |
| 14 | The Z key.                |         |       |      |      |       |
| 15 | <b>bbbbbbbb)ToString</b>  |         |       |      |      |       |
| 16 |                           |         |       |      |      |       |
| 17 | [C#]                      | public  | const |      | Keys | Zoom; |
| 18 | [C++]                     | public: | const |      | Keys | Zoom; |
| 19 | [VB]                      | Public  | Const | Zoom | As   | Keys  |
| 20 | [JScript]                 | public  | var   | Zoom | :    | Keys; |
| 21 |                           |         |       |      |      |       |
| 22 | <i>Description</i>        |         |       |      |      |       |
| 23 | The ZOOM key.             |         |       |      |      |       |
| 24 |                           |         |       |      |      |       |
| 25 |                           |         |       |      |      |       |

KeysConverter class (System.Windows.Forms)

a) *ToString*

*Description*

Provides a **System.ComponentModel.TypeConverter** to convert **System.Windows.Forms.Keys** objects to and from other representations.

b) *KeysConverter*

*Example Syntax:*

c) *ToString*

[C#] public KeysConverter();

[C++] public: KeysConverter();

[VB] Public Sub New()

[JScript] public function KeysConverter();

d) *CanConvertFrom*

[C#] public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType);

[C++] public: bool CanConvertFrom(ITypeDescriptorContext\* context, Type\* sourceType);

[VB] Overrides Public Function CanConvertFrom(ByVal context As ITypeDescriptorContext, ByVal sourceType As Type) As Boolean

[JScript] public override function CanConvertFrom(context : ITypeDescriptorContext, sourceType : Type) : Boolean; Geta a value indicating

whether this converter can convert an object in the Specified source type to the native type of the converter.

#### *Description*

Gets a value indicating whether this converter can convert an object in the specified source type to the native type of the converter using the specified context.

*Return Value:* **true** if this object can perform the conversion; otherwise, **false** .

Override this method to provide your own conversion requirements. An *ITypeDescriptorContext* that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null . The type to convert from.

#### *e) Compare*

[C#] public int Compare(object a, object b);

[C++] public: \_\_sealed int Compare(Object\* a, Object\* b);

[VB] NotOverridable Public Function Compare(ByVal a As Object, ByVal b As Object) As Integer

[JScript] public function Compare(a : Object, b : Object) : int;

#### *Description*

Compares two key values for equivalence.

*Return Value:* An integer indicating the relationship between the two comparands.

This method uses **System.String.Compare(System.String, System.String)** to compare the two objects. An **System.Object** that represents the first key to compare. An **System.Object** that represents the second key to compare.

*f) ConvertFrom*

```
[C#] public override object ConvertFrom(ITypeDescriptorContext context,
CultureInfo culture, object value);
[C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
CultureInfo* culture, Object* value);
[VB] Overrides Public Function ConvertFrom(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
As Object
[JavaScript] public override function ConvertFrom(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object) : Object; Converts the specified object to the
native type of the converter.
```

*Description*

Converts the specified object to the converter's native type.

*Return Value:* An Object that represents the converted *value* .

The context parameter can be used to extract additional information about the environment this converter is being invoked from. This may be **null** , so you should always check. Also, properties on the context object may also return null . An ITypeDescriptorContext that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null . A CultureInfo object to provide locale information. The object to convert.

*g) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
```



```

1 culture, Object* value, Type* destinationType);
2 [VB] Overrides Public Function ConvertTo(ByVal context As
3 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
4 ByVal destinationType As Type) As Object
5 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,
6 culture : CultureInfo, value : Object, destinationType : Type) : Object; Converts
7 the Specified object object to the specified destination type.

```

#### 9 *Description*

10 Converts the specified object to the specified destination type.

*Return Value:* An Object that represents the converted *value* .

11 Override this method to provide your own conversion requirements. An  
12 ITypeDescriptorContext that provides a format context, which can be used to  
13 extract additional information about the environment this converter is being  
14 invoked from. This parameter or properties of this parameter can be null . A  
15 CultureInfo object to provide locale information. The object to convert. The type  
16 to convert the object to.

#### 15 *h) GetStandardValues*

```

17 [C#] public override StandardValuesCollection
18 GetStandardValues(ITypeDescriptorContext context);
19 [C++] public: StandardValuesCollection*
20 GetStandardValues(ITypeDescriptorContext* context);
21 [VB] Overrides Public Function GetStandardValues(ByVal context As
22 ITypeDescriptorContext) As StandardValuesCollection
23 [JScript] public override function GetStandardValues(context :
24 ITypeDescriptorContext) : StandardValuesCollection; Returns a collection of

```

standard values for the data type that this type converter is designed for.

### *Description*

Returns a collection of standard values for the data type that this type converter is designed for when provided with a format context.

*Return Value:* A StandardValuesCollection that holds a standard set of valid values, or null if the data type does not support a standard set of values.

The collection returned contains the values of the keys that can be converted. An ITypeDescriptorContext that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null .

#### *i) GetStandardValuesExclusive*

```
[C#] public override bool GetStandardValuesExclusive(ITypeDescriptorContext
context);
```

```
[C++] public: bool GetStandardValuesExclusive(ITypeDescriptorContext*
context);
```

```
[VB] Overrides Public Function GetStandardValuesExclusive(ByVal context As
ITypeDescriptorContext) As Boolean
```

```
[JScript] public override function GetStandardValuesExclusive(context :
ITypeDescriptorContext) : Boolean;
```

### *Description*

Determines if the list of standard values returned from GetStandardValues is an exclusive list. If the list is exclusive, then no other values are valid, such as in an enum data type. If the list is not exclusive, then there are other valid values besides the list of standard values GetStandardValues provides.

*Return Value:* True if the collection returned from GetStandardValues is an exhaustive list of possible values, or false if other values are possible. The default for this method always returns false; Determines if the list of standard values returned from GetStandardValues is an exclusive list. If the list is

exclusive, then no other values are valid, such as in an enum data type. If the list is not exclusive, then there are other valid values besides the list of standard values `GetStandardValues` provides. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null.

j) *GetStandardValuesSupported*

[C#] public override bool GetStandardValuesSupported(ITypeDescriptorContext context);

[C++] public: bool GetStandardValuesSupported(ITypeDescriptorContext\* context);

[VB] Overrides Public Function GetStandardValuesSupported(ByVal context As ITypeDescriptorContext) As Boolean

[JScript] public override function GetStandardValuesSupported(context : ITypeDescriptorContext) : Boolean; Gets a value indicating whether this object supports a standard set of values that can be picked from a list.

*Description*

Gets a value indicating whether this object supports a standard set of values that can be picked from a list.

**Return Value:** Always returns **true**. An `ITypeDescriptorContext` that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null.

Label class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a standard Windows label.

**System.Windows.Forms.Label** controls are typically used to provide descriptive text for a control. For example, you can use a **System.Windows.Forms.Label** to add descriptive text for a **System.Windows.Forms.TextBox** control to inform the user about the type of data expected in the control.

*b) Label*

*Example Syntax:*

*c) ToString*

|           |         |                   |
|-----------|---------|-------------------|
| [C#]      | public  | Label();          |
| [C++]     | public: | Label();          |
| [VB]      | Public  | Sub New()         |
| [JScript] | public  | function Label(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.Label** class.

By default, a label is displayed with its **System.Windows.Forms.Label.AutoSize** property set to **false** and with its **System.Windows.Forms.Label.BorderStyle** property set to **BorderStyle.None** .

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoSize*
- l) *ToString*

#### *Description*

Gets or sets a value indicating whether the control is automatically resized to display its entire contents.

When this property is set to **true**, the control's height and width are automatically adjusted to display the entire contents of the control. This property is typically set to **true** when you use a **System.Windows.Forms.Label** control to display various lengths of text, such as the status of an application process. You can also use this property when the application will display text in various languages, and the size of the text may increase or decrease based on the language settings in Windows.

- m) *BackColor*
- n) *BackgroundImage*
- o) *ToString*

#### *Description*

Gets or sets the image rendered on the background of the control.

*p)      **BindingContext***

*q)      **BorderStyle***

*r)      **ToString***

### *Description*

Gets or sets the border style for the control.

You can use this property to add a border to the control. This property is typically used to differentiate a **System.Windows.Forms.Label** that labels another control from a **System.Windows.Forms.Label** that displays the status of a process in an application.

- s) *Bottom*
- t) *Bounds*
- u) *CanFocus*
- v) *CanSelect*
- w) *Capture*
- x) *CausesValidation*
- y) *ClientRectangle*
- z) *ClientSize*
- aa) *CompanyName*
- bb) *Container*
- cc) *ContainsFocus*
- dd) *ContextMenu*
- ee) *Controls*
- ff) *Created*
- gg) *CreateParams*
- hh) *ToString*

## *Description*

Overrides Control. A Label is a Win32 STATIC control, which we setup here.

- ii) *Cursor*
- jj) *DataBindings*
- kk) *DefaultImeMode*
- ll) *ToString*

### *Description*

Gets the default Input Method Editor(IME) mode supported by this control.

mm) *DefaultSize*

nn) *ToString*

[C#]       protected       override       Size       DefaultSize       {get;}

[C++]     protected:     \_\_property     virtual     Size     get\_DefaultSize();

[VB]   Overrides   Protected   ReadOnly   Property   DefaultSize   As   Size

[JScript]     protected     function     get     DefaultSize()     :     Size;

### *Description*

Deriving classes can override this to configure a default size for their control. This is more efficient than setting the size in the control's constructor.



oo) *DesignMode*

pp) *DisplayRectangle*

qq) *Disposing*

rr) *Dock*

ss) *Enabled*

tt) *Events*

uu) *FlatStyle*

vv) *ToString*

#### *Description*

Gets or sets the flat style appearance of the label control.

This property has no effect in the derived class,  
**System.Windows.Forms.LinkLabel** .

ww) *Focused*  
 xx) *Font*  
 yy) *FontHeight*  
 zz) *ForeColor*  
 aaa) *Handle*  
 bbb) *HasChildren*  
 ccc) *Height*  
 ddd) *Image*  
 eee) *ToString*

#### *Description*

Gets or sets the image that is displayed on a **System.Windows.Forms.Label**.

The **System.Windows.Forms.Label.Image** property cannot be used at the same time as the **System.Windows.Forms.Label.ImageList** and **System.Windows.Forms.Label.ImageIndex** properties. When the **System.Windows.Forms.Label.Image** property is used to display an image, the **System.Windows.Forms.Label.ImageList** and **System.Windows.Forms.Label.ImageIndex** properties are automatically set to their default settings.

fff) *ImageAlign*

ggg) *ToString*

```

[C#]      public      ContentAlignment      ImageAlign      {get;      set;}
[C++] public: __property ContentAlignment get_ImageAlign();public: __property
void                                             set_ImageAlign(ContentAlignment);
[VB]      Public      Property      ImageAlign      As      ContentAlignment

```

[JScript] public function get ImageAlign() : ContentAlignment;public function set ImageAlign(ContentAlignment);

#### *Description*

Gets or sets the alignment of an image that is displayed in the control.

This property enables you to align an image within the boundaries of the **System.Windows.Forms.Label** control to ensure that the image is properly displayed. You can add an image to a **System.Windows.Forms.Label** using the **System.Windows.Forms.Label.Image** property or the **System.Windows.Forms.Label.ImageList** and **System.Windows.Forms.Label.ImageIndex** properties. Images displayed in the control cannot be stretched or shrunk to fill the control if the control is larger or smaller than the image.

*hhh) ImageIndex*

*iii) ToString*

[C#]            public            int            ImageIndex            {get;            set;}

[C++]   public:   \_\_property   int   get\_ImageIndex();public:   \_\_property   void  
set\_ImageIndex(int);

[VB]            Public            Property            ImageIndex            As            Integer

[JScript]   public   function   get   ImageIndex()   :   int;public   function   set  
ImageIndex(int);

#### *Description*

Gets or sets the index value of the image displayed on the **System.Windows.Forms.Label**.

The **System.Windows.Forms.Label.ImageIndex** and the **System.Windows.Forms.Label.ImageList** properties cannot be used at the same time as the **System.Windows.Forms.Label.Image** property. When the **System.Windows.Forms.Label.ImageIndex** property and

**System.Windows.Forms.Label.ImageList** properties are used to display an image, the **System.Windows.Forms.Label.Image** property is automatically set to **null** .

*jjj) ImageList*

*kkk) ToString*

```
[C#]      public      ImageList      ImageList      {get;      set;}
```

```
[C++] public: __property ImageList* get_ImageList();public: __property void  
set_ImageList(ImageList*);
```

```
[VB]      Public      Property      ImageList      As      ImageList
```

```
[JScript] public function get ImageList() : ImageList;public function set  
ImageList(ImageList);
```

### *Description*

Gets or sets the **System.Windows.Forms.ImageList** that contains the images to display in the **System.Windows.Forms.Label** control.

The **System.Windows.Forms.Label.ImageIndex** and the **System.Windows.Forms.Label.ImageList** properties cannot be used at the same time as the **System.Windows.Forms.Label.Image** property. When the **System.Windows.Forms.Label.ImageIndex** property and **System.Windows.Forms.Label.ImageList** properties are used to display an image, the **System.Windows.Forms.Label.Image** property is set to **null** .

*lll) ImeMode*

*mmm) ToString*

```
[C#]      public      new      ImeMode      ImeMode      {get;      set;}
```

```
[C++] public: __property ImeMode get_ImeMode();public: __property void  
set_ImeMode(ImeMode);
```

```

1 [VB]      Public      Property      ImeMode      As      ImeMode
2 [JScript] public function get ImeMode() : ImeMode;public function set
3 ImeMode(ImeMode);
4

```

#### *Description*

Gets or sets the Input Method Editor(IME) mode supported by this control.

*nnn) InvokeRequired*

*ooo) IsAccessible*

*ppp) IsDisposed*

*qqq) IsHandleCreated*

*rrr) Left*

*sss) Location*

*ttt) Name*

*uuu) Parent*

*vvv) PreferredHeight*

*www) ToString*

#### *Description*

Gets the preferred height of the control.

This property returns the height that the control should be in order to properly display text, based on the font assigned to the control. You can use this property along with the **System.Windows.Forms.Label.PreferredWidth** property to ensure that the text in the **System.Windows.Forms.Label** control is displayed properly. You can use the **System.Windows.Forms.Label.AutoSize** property to automatically adjust the height and the width of the **System.Windows.Forms.Label** control, based on the text and font size.

xxx) *PreferredWidth*

yyy) *ToString*

```
[C#]      public      virtual      int      PreferredWidth      {get;}
[C++]     public:     __property  virtual  int      get_PreferredWidth();
[VB]      Overridable Public ReadOnly Property PreferredWidth As Integer
[JScript] public      function    get      PreferredWidth()      :      int;
```

### *Description*

Gets the preferred width of the control.

This property returns the length of the text string, but does not take line wrapping into consideration. For example, a text string that measures 300 pixels wide could be displayed as three lines in a **System.Windows.Forms.Label** that is only 100 pixels wide. The

**System.Windows.Forms.Label.PreferredWidth** property still returns 300 pixels. You can use this property, along with the

**System.Windows.Forms.Label.PreferredHeight** property, to ensure that the text in the **System.Windows.Forms.Label** control is displayed properly.

You can use the **System.Windows.Forms.Label.AutoSize** property to automatically adjust the height and the width of the

**System.Windows.Forms.Label** control based on the text and font size.

zzz) *ProductName*

aaaa) *ProductVersion*

bbbb) *RecreatingHandle*

cccc) *Region*

dddd) *RenderRightToLeft*

eeee) *RenderTransparent*

ffff) *ToString*

*Description*

Indicates whether the container control background is rendered on the  
**System.Windows.Forms.Label** .

*gggg) ResizeRedraw*

*hhhh) Right*

*iiii) RightToLeft*

*jjjj) ShowFocusCues*

*kkkk) ShowKeyboardCues*

*llll) Site*

*mmmm)Size*

*nnnn) TabIndex*

*oooo) TabStop*

*pppp) ToString*

#### *Description*

Gets or sets a value indicating whether the user can tab to the **System.Windows.Forms.Label** .

*qqqq) Tag*

*rrrr) Text*

*ssss) TextAlign*

*tttt) ToString*

#### *Description*

Gets or sets the alignment of text in the control.

You can use this property to align the text within a **System.Windows.Forms.Label** to match the layout of text and control on



your form. For example, if your controls are all located near the right edge of the form, you can set the **System.Windows.Forms.Label.TextAlign** property to **ContentAlignment.Right** and the text will be aligned with the right edge of the control to align with your controls.

*uuuu) Top*

*vvvv) TopLevelControl*

*www)UseMnemonic*

*xxxx) ToString*

#### *Description*

Gets or sets a value indicating whether the control interprets an ampersand character (&) in the control's **System.Windows.Forms.Control.Text** property to be an access key prefix character.

If the **System.Windows.Forms.Label.UseMnemonic** property is set to **true** and a mnemonic character (a character preceded by the ampersand) is defined in the **System.Windows.Forms.Control.Text** property of the **System.Windows.Forms.Label**, pressing ALT+ the mnemonic character sets the focus to the control that follows the **System.Windows.Forms.Label** in the tab order. You can use this property to provide proper keyboard navigation to the controls on your form.

*yyyy) Visible*

*zzzz) Width*

*aaaaa) WindowTarget*

*bbbb) ToString*

#### *Description*

Occurs when the value of the **System.Windows.Forms.Label.AutoSize** property has changed.

For more information about handling events, see .

*cccc) ToString*

*dddd) ToString*

*eeee) ToString*

*ffff) ToString*

#### *Description*

Occurs when the value of the **System.Windows.Forms.Label.TextAlign** property has changed.

For more information about handling events, see .

*gggg) CalcImageRenderBounds*

[C#] protected Rectangle CalcImageRenderBounds(Image image, Rectangle r, ContentAlignment align);

[C++] protected: Rectangle CalcImageRenderBounds(Image\* image, Rectangle r, ContentAlignment align);

[VB] Protected Function CalcImageRenderBounds(ByVal image As Image, ByVal r As Rectangle, ByVal align As ContentAlignment) As Rectangle

[JScript] protected function CalcImageRenderBounds(image : Image, r : Rectangle, align : ContentAlignment) : Rectangle;

#### *Description*

Determines the size and location of an image drawn within the **System.Windows.Forms.Label** control based on the alignment of the control.

*Return Value:* A **System.Drawing.Rectangle** that represents the size and location of the specified image within the control.

You can use this method within a derived class of **System.Windows.Forms.Label**, to determine the size and location of an image to draw within the **System.Windows.Forms.Label** control based on its location within the control. The location of the image is based on the value of the control's **System.Windows.Forms.Label.ImageAlign** property. The **System.Drawing.Image** used to determine size and location when drawn within the control. A **System.Drawing.Rectangle** that represents the area to draw the image in. The alignment of content within the control.

#### *hhhhh)CreateAccessibilityInstance*

```
[C#]    protected    override    AccessibleObject    CreateAccessibilityInstance();
[C++]    protected:    AccessibleObject*    CreateAccessibilityInstance();
[VB]    Overrides    Protected    Function    CreateAccessibilityInstance()    As
AccessibleObject
[JScript]    protected    override    function    CreateAccessibilityInstance()    :
AccessibleObject;
```

#### *Description*

#### *iiii) Dispose*

```
[C#]    protected    override    void    Dispose(bool    disposing);
[C++]    protected:    void    Dispose(bool    disposing);
[VB]    Overrides    Protected    Sub    Dispose(ByVal    disposing    As    Boolean)
[JScript]    protected    override    function    Dispose(disposing : Boolean);
```

#### *jjjj) DrawImage*

```
[C#]    protected    void    DrawImage(Graphics    g, Image    image, Rectangle    r,
```

```

1 ContentAlignment                                align);
2 [C++] protected: void DrawImage(Graphics* g, Image* image, Rectangle r,
3 ContentAlignment                                align);
4 [VB] Protected Sub DrawImage(ByVal g As Graphics, ByVal image As Image,
5 ByVal r As Rectangle, ByVal align As ContentAlignment)
6 [JScript] protected function DrawImage(g : Graphics, image : Image, r :
7 Rectangle, align : ContentAlignment);
8

```

### *Description*

Draws an **System.Drawing.Image** within the specified bounds.

When you are creating your own control that derives from **System.Windows.Forms.Label**, you can use this method to draw images onto the surface of the control. The **System.Drawing.Graphics** surface on which to draw. The **System.Drawing.Image** to draw. The **System.Drawing.Rectangle** bounds to draw within. The alignment of the image to draw within the **System.Windows.Forms.Label**.

### *kkkkk) OnAutoSizeChanged*

```

16
17 [C#] protected virtual void OnAutoSizeChanged(EventArgs e);
18 [C++] protected: virtual void OnAutoSizeChanged(EventArgs* e);
19 [VB] Overridable Protected Sub OnAutoSizeChanged(ByVal e As EventArgs)
20 [JScript] protected function OnAutoSizeChanged(e : EventArgs);
21

```

### *Description*

Raises the **System.Windows.Forms.Label.AutoSizeChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.EventArgs** that contains the event data.

**lllll) OnEnabledChanged**

[C#] protected override void OnEnabledChanged(EventArgs e);  
[C++] protected: void OnEnabledChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnEnabledChanged(ByVal e As EventArgs)  
[JScript] protected override function OnEnabledChanged(e : EventArgs);

**mmmmm)OnFontChanged**

[C#] protected override void OnFontChanged(EventArgs e);  
[C++] protected: void OnFontChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)  
[JScript] protected override function OnFontChanged(e : EventArgs);

**nnnnn)OnPaint**

[C#] protected override void OnPaint(PaintEventArgs e);  
[C++] protected: void OnPaint(PaintEventArgs\* e);  
[VB] Overrides Protected Sub OnPaint(ByVal e As PaintEventArgs)  
[JScript] protected override function OnPaint(e : PaintEventArgs);

**Description**

**ooooo) OnParentChanged**

[C#] protected override void OnParentChanged(EventArgs e);  
[C++] protected: void OnParentChanged(EventArgs\* e);

1 [VB] Overrides Protected Sub OnParentChanged(ByVal e As EventArgs)

2 [JScript] protected override function OnParentChanged(e : EventArgs);

3 *ppppp) OnTextAlignChanged*

4  
5 [C#] protected virtual void OnTextAlignChanged(EventArgs e);

6 [C++] protected: virtual void OnTextAlignChanged(EventArgs\* e);

7 [VB] Overridable Protected Sub OnTextAlignChanged(ByVal e As EventArgs)

8 [JScript] protected function OnTextAlignChanged(e : EventArgs);

9  
10 *Description*

11 Raises the **System.Windows.Forms.Label.TextAlignChanged** event.

12 Raising an event invokes the event handler through a delegate. For more  
13 information, see . An **System.EventArgs** that contains the event data.

14 *qqqqq) OnTextChanged*

15 [C#] protected override void OnTextChanged(EventArgs e);

16 [C++] protected: void OnTextChanged(EventArgs\* e);

17 [VB] Overrides Protected Sub OnTextChanged(ByVal e As EventArgs)

18 [JScript] protected override function OnTextChanged(e : EventArgs);

19  
20 *rrrrr) OnVisibleChanged*

21  
22 [C#] protected override void OnVisibleChanged(EventArgs e);

23 [C++] protected: void OnVisibleChanged(EventArgs\* e);

24 [VB] Overrides Protected Sub OnVisibleChanged(ByVal e As EventArgs)

25 [JScript] protected override function OnVisibleChanged(e : EventArgs);

*sssss) ProcessMnemonic*

[C#] protected override bool ProcessMnemonic(char charCode);  
[C++] protected: bool ProcessMnemonic(\_\_wchar\_t charCode);  
[VB] Overrides Protected Function ProcessMnemonic(ByVal charCode As Char)  
As Boolean  
[JScript] protected override function ProcessMnemonic(charCode : Char) :  
Boolean;

*Description*

Overrides Control. This is called when the user has pressed an Alt-CHAR key combination and determines if that combination is an interesting mnemonic for this control.

*ttttt) SetBoundsCore*

[C#] protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);  
[C++] protected: void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified);  
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As BoundsSpecified)  
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int, height : int, specified : BoundsSpecified);

*Description*

Overrides Control.setBoundsCore to enforce autoSize.

*uuuuu)ToString*

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

*Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

*vvvvv) WndProc*

|           |            |           |          |                             |
|-----------|------------|-----------|----------|-----------------------------|
| [C#]      | protected  | override  | void     | WndProc(ref Message m);     |
| [C++]     | protected: |           | void     | WndProc(Message* m);        |
| [VB]      | Overrides  | Protected | Sub      | WndProc(ByRef m As Message) |
| [JScript] | protected  | override  | function | WndProc(m : Message);       |

*Description*

Overrides Control. This processes certain messages that the Win32 STATIC class would normally override. The message to process.

LabelEditEventArgs class (System.Windows.Forms)

*a) WndProc*

*Description*



Provides data for the **System.Windows.Forms.ListView.LabelEdit** event.

A **System.Windows.Forms.LabelEditEventArgs** specifies the index and caption of a **System.Windows.Forms.ListViewItem** and the caption after it has been edited by the user. This class also provides the **System.Windows.Forms.LabelEditEventArgs.CancelEdit** property, which enables code in an event handler for the **System.Windows.Forms.ListView.LabelEdit** event to cancel the changes made to the label by the user.

*b) LabelEditEventArgs*

*Example Syntax:*

*c) WndProc*

[C#]                    public                    LabelEditEventArgs(int                    item);  
[C++]                    public:                    LabelEditEventArgs(int                    item);  
[VB]                    Public                    Sub                    New(ByVal                    item                    As                    Integer)  
[JScript] public function LabelEditEventArgs(item : int); Initializes a new instance of the **System.Windows.Forms.LabelEditEventArgs** class.

#### *Description*

Initializes a new instance of the **System.Windows.Forms.LabelEditEventArgs** class with the specified index to the **System.Windows.Forms.ListViewItem** to edit.

You can use this constructor when raising the **System.Windows.Forms.ListView.LabelEdit** event at run time to specify a specific list item in the **System.Windows.Forms.ListView** to edit. The zero-based index of the **System.Windows.Forms.ListViewItem**, containing the label to edit.

*d) LabelEditEventArgs*

*Example Syntax:*

e) *WndProc*

```
[C#]      public      LabelEventArgs(int      item,      string      label);  
[C++]     public:     LabelEventArgs(int      item,      String*      label);  
[VB]      Public Sub New(ByVal item As Integer, ByVal label As String)  
[JScript] public function LabelEventArgs(item : int, label : String);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.LabelEventArgs** class with the specified index to the **System.Windows.Forms.ListViewItem** being edited and the new text for the label of the **System.Windows.Forms.ListViewItem** .

You can use this constructor when raising the **System.Windows.Forms.ListView.LabelEdit** event at run time to specify a list item in the **System.Windows.Forms.ListView** to edit. The constructor also enables you to specify the new text associated with the label of the **System.Windows.Forms.ListViewItem** . The zero-based index of the **System.Windows.Forms.ListViewItem**, containing the label to edit. The new text assigned to the label of the **System.Windows.Forms.ListViewItem**.

f) *CancelEdit*

g) *WndProc*

```
[C#]      public      bool      CancelEdit      {get;      set;}  
[C++]     public:     __property bool get_CancelEdit();public: __property void  
set_CancelEdit(bool);  
[VB]      Public      Property      CancelEdit      As      Boolean  
[JScript] public function get CancelEdit() : Boolean;public function set  
CancelEdit(Boolean);
```

*Description*

Gets or sets a value indicating whether changes made to the label of the **System.Windows.Forms.ListViewItem** should be canceled.

You can use this property to cancel changes made to the label of a **System.Windows.Forms.ListViewItem** and revert it to its original text. Use this property to prevent an item's caption from being changed to a value that does not meet your application's requirements.

*h) Item*

*i) WndProc*

|           |         |            |          |                 |
|-----------|---------|------------|----------|-----------------|
| [C#]      | public  | int        | Item     | {get;}          |
| [C++]     | public: | __property | int      | get_Item();     |
| [VB]      | Public  | ReadOnly   | Property | Item As Integer |
| [JScript] | public  | function   | get      | Item() : int;   |

*Description*

Gets the zero-based index of the **System.Windows.Forms.ListViewItem** containing the label to edit.

*j) Label*

*k) WndProc*

|           |         |            |          |                   |
|-----------|---------|------------|----------|-------------------|
| [C#]      | public  | string     | Label    | {get;}            |
| [C++]     | public: | __property | String*  | get_Label();      |
| [VB]      | Public  | ReadOnly   | Property | Label As String   |
| [JScript] | public  | function   | get      | Label() : String; |

1  
2 *Description*

3 Gets the new text assigned to the label of the  
4 **System.Windows.Forms.ListViewItem** .

5 LabelEditEventHandler delegate (System.Windows.Forms)

6 *a) ToString*

7  
8  
9 *Description*

10 Represents the method that handles the  
11 **System.Windows.Forms.ListView.LabelEdit** event of a  
12 **System.Windows.Forms.ListView** . The source of the event. A  
13 **System.Windows.Forms.LabelEditEventArgs** that contains the event data.

14 When you create a **System.Windows.Forms.LabelEditEventHandler**  
15 delegate, you identify the method that will handle the event. To associate the  
16 event with your event handler, add an instance of the delegate to the event. The  
17 event handler is called whenever the event occurs, unless you remove the  
18 delegate. For more information about delegates, see .

19 LayoutEventArgs class (System.Windows.Forms)

20 *a) ToString*

21  
22  
23 *Description*

24 Provides data for the **System.Windows.Forms.Control.Layout** event.

25 Changes to a control such as resizing, showing or hiding child controls, and  
adding or removing child controls make it necessary for a control to layout its  
child controls. A **System.Windows.Forms.LayoutEventArgs** object specifies  
the child control that has been changed, and the child control's affected  
property. For example, if a control has been made visible since the last layout  
operation, the **System.Windows.Forms.Control.Visible** property is affected.

**b) *LayoutEventArgs***

*Example Syntax:*

**c) *ToString***

[C#] public LayoutEventArgs(Control affectedControl, string affectedProperty);

[C++] public: LayoutEventArgs(Control\* affectedControl, String\* affectedProperty);

[VB] Public Sub New(ByVal affectedControl As Control, ByVal affectedProperty As String)

[JScript] public function LayoutEventArgs(affectedControl : Control, affectedProperty : String);

*Description*

Initializes a new instance of the **System.Windows.Forms.LayoutEventArgs** class with the specified control and property affected. The **System.Windows.Forms.Control** affected by the change. The property affected by the change.

**d) *AffectedControl***

**e) *ToString***

[C#] public Control AffectedControl {get;}

[C++] public: \_\_property Control\* get\_AffectedControl();

[VB] Public ReadOnly Property AffectedControl As Control

[JScript] public function get AffectedControl() : Control;

*Description*

Gets the child control affected by the change.

*f) AffectedProperty*

*g) ToString*

```
[C#]      public      string      AffectedProperty      {get;}
```

```
[C++]      public:      __property      String*      get_AffectedProperty();
```

```
[VB]      Public      ReadOnly      Property      AffectedProperty      As      String
```

```
[JScript]      public      function      get      AffectedProperty()      :      String;
```

#### *Description*

Gets the property affected by the change.

If a child control has been made visible since the last layout operation, the **System.Windows.Forms.Control.Visible** property is affected.

LayoutEventHandler delegate (System.Windows.Forms)

*a) ToString*

#### *Description*

Represents the method that will handle the **System.Windows.Forms.Control.Layout** event of a **System.Windows.Forms.Control** . The source of the event. A **System.Windows.Forms.LayoutEventArgs** that contains the event data.

When you create a **System.Windows.Forms.LayoutEventArgs** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

LeftRightAlignment enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies whether an object or text is aligned to the left or right of a reference point.

Use the members of this enumeration to set the values of the **System.Windows.Forms.DateTimePicker.DropDownAlign** and **System.Windows.Forms.UpDownBase.UpDownAlign** properties.

*b) ToString*

|           |         |       |                    |                       |
|-----------|---------|-------|--------------------|-----------------------|
| [C#]      | public  | const | LeftRightAlignment | Left;                 |
| [C++]     | public: | const | LeftRightAlignment | Left;                 |
| [VB]      | Public  | Const | Left               | As LeftRightAlignment |
| [JScript] | public  | var   | Left               | : LeftRightAlignment; |

*Description*

The object or text is aligned to the left of the reference point.

*c) ToString*

|           |         |       |                    |                       |
|-----------|---------|-------|--------------------|-----------------------|
| [C#]      | public  | const | LeftRightAlignment | Right;                |
| [C++]     | public: | const | LeftRightAlignment | Right;                |
| [VB]      | Public  | Const | Right              | As LeftRightAlignment |
| [JScript] | public  | var   | Right              | : LeftRightAlignment; |

## Description

The object or text is aligned to the right of the reference point.

LinkLabel.Link class (System.Windows.Forms)

### a) ToString

## Description

Represents a link within a **System.Windows.Forms.LinkLabel** control.

The **System.Windows.Forms.LinkLabel.Link** class defines the properties of a link within a **System.Windows.Forms.LinkLabel** control. You can use these properties to provide data to the

**System.Windows.Forms.LinkLabel.LinkClicked** event of the **System.Windows.Forms.LinkLabel** control to perform tasks when the link is clicked in the control. The

**System.Windows.Forms.LinkLabel.Link.LinkData** property enables you to define information that the **System.Windows.Forms.LinkLabel.LinkClicked** event can use to display a URL within Microsoft Internet Explorer or to open a file.

### b) Enabled

### c) ToString

```
[C#] public bool Enabled {get; set;}
```

```
[C++] public: __property bool get_Enabled();public: __property void  
set_Enabled(bool);
```

```
[VB] Public Property Enabled As Boolean
```

```
[JScript] public function get Enabled() : Boolean;public function set  
Enabled(Boolean);
```



## Description

Gets or sets a value indicating whether the link is enabled.

You can use this property to display a link in a disabled state within the **System.Windows.Forms.LinkLabel** control. When a link is disabled, clicking on the link does not cause the **System.Windows.Forms.LinkLabel** control to raise the **System.Windows.Forms.LinkLabel.LinkClicked** event.

d) *Length*

e) *ToString*

```
[C#]          public          int          Length          {get;          set;}
```

```
[C++]  public:  __property  int  get_Length();public:  __property  void  
set_Length(int);
```

```
[VB]          Public          Property          Length          As          Integer
```

```
[JScript] public function get Length() : int;public function set Length(int);
```

## Description

Gets or sets the number of characters in the link text.

To specify text from the **System.Windows.Forms.LinkLabel** to display as a link, set the **System.Windows.Forms.LinkLabel.Link.Start** property to the location in the text to start creating the link. After the **System.Windows.Forms.LinkLabel.Link.Start** property is set, set the value of the **System.Windows.Forms.LinkLabel.Link.Length** property to the number of characters, including the character position specified in the **System.Windows.Forms.LinkLabel.Link.Start** property, that you want to make the link text. For example, if you want to make the first word of the text "The quick brown fox" a link, you set the **System.Windows.Forms.LinkLabel.Link.Start** property to zero (0) and the **System.Windows.Forms.LinkLabel.Link.Length** property to three (3).

f) *LinkData*

g) *ToString*

[C#] public object LinkData {get; set;}

[C++] public: \_\_property Object\* get\_LinkData();public: \_\_property void set\_LinkData(Object\*);

[VB] Public Property LinkData As Object

[JScript] public function get LinkData() : Object;public function set LinkData(Object);

### *Description*

Gets or sets the data associated with the link.

You can use this property to supply information related to the link. The information provided by this property can be used within the **System.Windows.Forms.LinkLabel.LinkClicked** event of the **System.Windows.Forms.LinkLabel** to provide information about the link that can be used to process the link being clicked. For example, you can specify the URL to display in Internet Explorer when the link is clicked as the value of the **System.Windows.Forms.LinkLabel.Link.LinkData** property. You can also use the **System.Windows.Forms.LinkLabel.Link.LinkData** property to identify a dialog to display when the user clicks on the link.

h) *Start*

i) *ToString*

[C#] public int Start {get; set;}

[C++] public: \_\_property int get\_Start();public: \_\_property void set\_Start(int);

[VB] Public Property Start As Integer

[JScript] public function get Start() : int;public function set Start(int);

## Description

Gets or sets the starting location of the link within the text of the **System.Windows.Forms.LinkLabel** .

To specify text from the **System.Windows.Forms.LinkLabel** to display as a link, set the **System.Windows.Forms.LinkLabel.Link.Start** property to the location in the text to start creating the link. After the **System.Windows.Forms.LinkLabel.Link.Start** property is set, set the value of the **System.Windows.Forms.LinkLabel.Link.Length** property to the number of characters, including the character position specified in the **System.Windows.Forms.LinkLabel.Link.Start** property, that you want to make the link text. For example, if you want to make the first word of the text "The quick brown fox" a link, you set the **System.Windows.Forms.LinkLabel.Link.Start** property to zero (0) and the **System.Windows.Forms.LinkLabel.Link.Length** property to three (3).

j) *Visited*

k) *ToString*

```
[C#]          public          bool          Visited          {get;          set;}
[C++] public:  __property bool  get_Visited();public:  __property void
set_Visited(bool);
[VB]          Public          Property          Visited          As          Boolean
[JScript] public function get Visited() : Boolean;public function set
Visited(Boolean);
```

## Description

Gets or sets a value indicating whether the user has visited the link.

A **System.Windows.Forms.LinkLabel** control does not automatically denote that a link is a visited link. To display the link as a visited link, you can set the value of this property to **true** in an event handler for the **System.Windows.Forms.LinkLabel.LinkClicked** event of a **System.Windows.Forms.LinkLabel** . A visited link is displayed using the color

specified in the **System.Windows.Forms.LinkLabel.VisitedLinkColor** property of the **System.Windows.Forms.LinkLabel** control. Once the form containing the **System.Windows.Forms.LinkLabel** control is closed, the all display state associated with the link is deleted. In order to retain the display state of the link, you need to store the display state of the link in a registry setting associated with your application.

LinkArea structure (System.Windows.Forms)

*a) ToString*

*Description*

Represents an area within a **System.Windows.Forms.LinkLabel** control that represents a hyperlink within the control.

There are two ways to add a hyperlink to the text of a **System.Windows.Forms.LinkLabel** control. You can access the **System.Windows.Forms.LinkLabel.LinkCollection.Add(System.Int32, System.Int32)** method of the **System.Windows.Forms.LinkLabel.LinkCollection** class through the **System.Windows.Forms.LinkLabel.Links** property of the **System.Windows.Forms.LinkLabel** to add multiple hyperlinks to the control's text. If you only need to add a single hyperlink to the text of the control, you can use the **System.Windows.Forms.LinkLabel.LinkArea** property of the **System.Windows.Forms.LinkLabel**. This property accepts a **System.Windows.Forms.LinkArea** object that defines the location of the hyperlink within the control's text. When a hyperlink is specified using the **System.Windows.Forms.LinkArea** property, the link area is then added to the **System.Windows.Forms.LinkLabel.LinkCollection** in the same manner as adding the link using the **System.Windows.Forms.LinkLabel.LinkCollection.Add(System.Int32, System.Int32)** method of the **System.Windows.Forms.LinkLabel.LinkCollection**.

*b) LinkArea*

*Example Syntax:*

c) *ToString*

```
[C#]      public      LinkArea(int      start,      int      length);
[C++]      public:      LinkArea(int      start,      int      length);
[VB] Public Sub New(ByVal start As Integer, ByVal length As Integer)
[JScript] public function LinkArea(start : int, length : int);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.LinkArea** class. The zero-based starting location of the link area within the text of the **System.Windows.Forms.LinkLabel**. The number of characters, after the starting character, to include in the link area.

d) *IsEmpty*

e) *ToString*

```
[C#]      public      bool      IsEmpty      {get;}
[C++]      public:      __property      bool      get_IsEmpty();
[VB] Public ReadOnly Property IsEmpty As Boolean
[JScript] public function get IsEmpty() : Boolean;
```

*Description*

Gets a value indicating whether the **System.Windows.Forms.LinkArea** is empty.

You can use this property to determine whether a valid link area has been specified in this object instead of evaluating the values of the **System.Windows.Forms.LinkArea.Length** and **System.Windows.Forms.LinkArea.Start** properties.

f) *Length*

g) *ToString*

[C#]            public            int            Length            {get;            set;}

[C++]   public:   \_\_property   int   get\_Length();public:   \_\_property   void  
set\_Length(int);

[VB]            Public            Property            Length            As            Integer

[JScript]   public   function   get   Length() : int;public   function   set   Length(int);

### *Description*

Gets or sets the number of characters in the link area.

To specify text from the **System.Windows.Forms.LinkLabel** to display as a link, set the **System.Windows.Forms.LinkArea.Start** property to the location in the text to start creating the link. After the **System.Windows.Forms.LinkArea.Start** property is set, set the value of the **System.Windows.Forms.LinkArea.Length** property to the number of characters, including the character position specified in the **System.Windows.Forms.LinkArea.Start** property, that you want to make the link text. For example, if you want to make the first word of the text "The quick brown fox" a link, you set the **System.Windows.Forms.LinkArea.Start** property to zero (0) and the **System.Windows.Forms.LinkArea.Length** property to three (3).

h) *Start*

i) *ToString*

[C#]            public            int            Start            {get;            set;}

[C++]   public:   \_\_property   int   get\_Start();public:   \_\_property   void   set\_Start(int);

[VB]            Public            Property            Start            As            Integer

[JScript]   public   function   get   Start() : int;public   function   set   Start(int);

## Description

Gets or sets the starting location of the link area within the text of the **System.Windows.Forms.LinkLabel**.

To specify text from the **System.Windows.Forms.LinkLabel** to display as a link, set the **System.Windows.Forms.LinkArea.Start** property to the location in the text to start creating the link. After the **System.Windows.Forms.LinkArea.Start** property is set, set the value of the **System.Windows.Forms.LinkArea.Length** property to the number of characters, including the character position specified in the **System.Windows.Forms.LinkArea.Start** property, that you want to make the link text. For example, if you want to make the first word of the text "The quick brown fox" a link, you set the **System.Windows.Forms.LinkArea.Start** property to zero (0) and the **System.Windows.Forms.LinkArea.Length** property to three (3).

### j) Equals

```
[C#]      public      override      bool      Equals(object      o);
[C++]      public:      bool      Equals(Object*      o);
[VB] Overrides Public Function Equals(ByVal o As Object) As Boolean
[JScript] public override function Equals(o : Object) : Boolean;
```

## Description

### k) GetHashCode

```
[C#]      public      override      int      GetHashCode();
[C++]      public:      int      GetHashCode();
[VB] Overrides Public Function GetHashCode() As Integer
[JScript] public override function GetHashCode() : int;
```

1  
2 *Description*

3  
4 LinkArea.LinkAreaConverter class (System.Windows.Forms)

5 *a) ToString*

6  
7  
8 *Description*

9 Provides a type converter to convert  
10 **System.Windows.Forms.LinkArea.LinkAreaConverter** objects to and from  
various other representations.

11 For more information about type converters, see the  
12 **System.ComponentModel.TypeConverter** base class and .

13 *b) LinkArea.LinkAreaConverter*

14 *Example Syntax:*

15 *c) ToString*

16  
17 [C#] public LinkArea.LinkAreaConverter();

18 [C++] public: LinkAreaConverter();

19 [VB] Public Sub New()

20 [JScript] public function LinkArea.LinkAreaConverter();

21 *d) CanConvertFrom*

22  
23 [C#] public override bool CanConvertFrom(ITypeDescriptorContext context,  
24 Type sourceType);

25 [C++] public: bool CanConvertFrom(ITypeDescriptorContext\* context, Type\*



1 sourceType);

2 [VB] Overrides Public Function CanConvertFrom(ByVal context As  
3 ITypeDescriptorContext, ByVal sourceType As Type) As Boolean  
4 [JScript] public override function CanConvertFrom(context :  
5 ITypeDescriptorContext, sourceType : Type) : Boolean;

6  
7 *Description*

8 Determines if this converter can convert an object in the given source type to the  
9 native type of the converter.

9 *Return Value:* True if this object can perform the conversion. A formatter  
10 context. This object can be used to extract additional information about the  
11 environment this converter is being invoked from. This may be null, so you  
12 should always check. Also, properties on the context object may also return null.  
13 The type you wish to convert from.

12 e) *CanConvertTo*

14 [C#] public override bool CanConvertTo(ITypeDescriptorContext context, Type  
15 destinationType);

16 [C++] public: bool CanConvertTo(ITypeDescriptorContext\* context, Type\*  
17 destinationType);

18 [VB] Overrides Public Function CanConvertTo(ByVal context As  
19 ITypeDescriptorContext, ByVal destinationType As Type) As Boolean  
20 [JScript] public override function CanConvertTo(context :  
21 ITypeDescriptorContext, destinationType : Type) : Boolean;

22  
23 *Description*

24 Gets a value indicating whether this converter can convert an object to the given  
25 destination type using the context.

*Return Value:* **true** if this converter can perform the conversion; otherwise, **false** .

The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null** , so always check. Also, properties on the context object can return **null** . An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you wish to convert to.

#### *f) ConvertFrom*

```
[C#] public override object ConvertFrom(ITypeDescriptorContext context,
CultureInfo culture, object value);
[C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
CultureInfo* culture, Object* value);
[VB] Overrides Public Function ConvertFrom(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
As Object
[JScript] public override function ConvertFrom(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object) : Object;
```

#### *Description*

Converts the given object to the converter's native type.

*Return Value:* The converted object. This will throw an exception if the conversion could not be performed. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null. An optional culture info. If not supplied the current culture is assumed. The object to convert.

#### *g) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
```

```

1 CultureInfo culture, object value, Type destinationType);
2 [C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
3 culture, Object* value, Type* destinationType);
4 [VB] Overrides Public Function ConvertTo(ByVal context As
5 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
6 ByVal destinationType As Type) As Object
7 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,
8 culture : CultureInfo, value : Object, destinationType : Type) : Object;
9

```

#### 10 *Description*

11 Converts the given object to another type. The most common types to convert  
12 are to and from a string object. The default implementation will make a call to  
13 ToString on the object if the object is valid and if the destination type is string. If  
14 this cannot convert to the destination type, this will throw a  
15 NotSupportedException.

16 *Return Value:* The converted object. A formatter context. This object can be  
17 used to extract additional information about the environment this converter is  
18 being invoked from. This may be null, so you should always check. Also,  
19 properties on the context object may also return null. An optional culture info. If  
20 not supplied the current culture is assumed. The object to convert. The type to  
21 convert the object to.

#### 22 *h) CreateInstance*

```

23 [C#] public override object CreateInstance(ITypeDescriptorContext context,
24 IDictionary propertyValues);
25 [C++] public: Object* CreateInstance(ITypeDescriptorContext* context,
26 IDictionary* propertyValues);
27 [VB] Overrides Public Function CreateInstance(ByVal context As
28 ITypeDescriptorContext, ByVal propertyValues As IDictionary) As Object

```

```

1 [JScript] public override function CreateInstance(context :
2 ITypeDescriptorContext, propertyValues : IDictionary) : Object;
3

```

#### 4 *Description*

5 Creates an instance of this type given a set of property values for the object.  
6 This is useful for objects that are immutable, but still want to provide changable  
7 properties.

8 *Return Value:* The newly created object, or null if the object could not be  
9 created. The default implementation returns null. A type descriptor through  
10 which additional context may be provided. A dictionary of new property values.  
11 The dictionary contains a series of name-value pairs, one for each property  
12 returned from GetProperties.

#### 10 i) *GetCreateInstanceSupported*

```

12 [C#] public override bool GetCreateInstanceSupported(ITypeDescriptorContext
13 context);

```

```

14 [C++] public: bool GetCreateInstanceSupported(ITypeDescriptorContext*
15 context);

```

```

16 [VB] Overrides Public Function GetCreateInstanceSupported(ByVal context As
17 ITypeDescriptorContext) As Boolean

```

```

18 [JScript] public override function GetCreateInstanceSupported(context :
19 ITypeDescriptorContext) : Boolean;
20

```

#### 21 *Description*

22 Determines if changing a value on this object should require a call to  
23 CreateInstance to create a new value.

24 *Return Value:* Returns true if CreateInstance should be called when a change is  
25 made to one or more properties of this object. A type descriptor through which  
additional context may be provided.

## j) *GetProperties*

```
[C#]      public      override      PropertyDescriptorCollection
GetProperties(ITypeDescriptorContext context, object value, Attribute[]
attributes);
```

```
[C++]      public:      PropertyDescriptorCollection*
GetProperties(ITypeDescriptorContext* context, Object* value, Attribute*
attributes[]);
```

```
[VB]  Overrides  Public  Function  GetProperties(ByVal context As
ITypeDescriptorContext, ByVal value As Object, ByVal attributes() As Attribute)
As
PropertyDescriptorCollection
```

```
[JScript] public override function GetProperties(context : ITypeDescriptorContext,
value : Object, attributes : Attribute[]) : PropertyDescriptorCollection;
```

### *Description*

Retrieves the set of properties for this type. By default, a type has does not return any properties. An easy implementation of this method can just call `TypeDescriptor.GetProperties` for the correct data type.

**Return Value:** The set of properties that should be exposed for this data type. If no properties should be exposed, this may return null. The default implementation always returns null. A type descriptor through which additional context may be provided. The value of the object to get the properties for.

## k) *GetPropertiesSupported*

```
[C#]  public  override  bool  GetPropertiesSupported(ITypeDescriptorContext
context);
```

```
[C++]  public:  bool  GetPropertiesSupported(ITypeDescriptorContext* context);
```

```
[VB]  Overrides  Public  Function  GetPropertiesSupported(ByVal context As
```

```
1 ITypeDescriptorContext) As Boolean
2 [JScript] public override function GetPropertiesSupported(context :
3 ITypeDescriptorContext) : Boolean;
```

*Description*

Determines if this object supports properties. By default, this is false.  
*Return Value:* Returns true if GetProperties should be called to find the properties of this object. A type descriptor through which additional context may be provided.

LinkBehavior enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the behaviors of a link in a **System.Windows.Forms.LinkLabel** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.LinkLabel.LinkBehavior** property of the **System.Windows.Forms.LinkLabel** .

*b) ToString*

```
19 [C#] public const LinkBehavior AlwaysUnderline;
20 [C++] public: const LinkBehavior AlwaysUnderline;
21 [VB] Public Const AlwaysUnderline As LinkBehavior
22 [JScript] public var AlwaysUnderline : LinkBehavior;
```

*Description*

The link always displays with underlined text.

c) *ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | LinkBehavior   | HoverUnderline; |
| [C++]     | public: | const | LinkBehavior   | HoverUnderline; |
| [VB]      | Public  | Const | HoverUnderline | As LinkBehavior |
| [JScript] | public  | var   | HoverUnderline | : LinkBehavior; |

*Description*

The link displays underlined text only when the mouse is hovered over the link text.

d) *ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | LinkBehavior   | NeverUnderline; |
| [C++]     | public: | const | LinkBehavior   | NeverUnderline; |
| [VB]      | Public  | Const | NeverUnderline | As LinkBehavior |
| [JScript] | public  | var   | NeverUnderline | : LinkBehavior; |

*Description*

The link text is never underlined. The link can still be distinguished from other text by use of the **System.Windows.Forms.LinkLabel.LinkColor** property of the **System.Windows.Forms.LinkLabel** control.

e) *ToString*

|       |         |       |               |                 |
|-------|---------|-------|---------------|-----------------|
| [C#]  | public  | const | LinkBehavior  | SystemDefault;  |
| [C++] | public: | const | LinkBehavior  | SystemDefault;  |
| [VB]  | Public  | Const | SystemDefault | As LinkBehavior |

1 [JScript]        public        var        SystemDefault        :        LinkBehavior;

2  
3 *Description*

4 The link is displayed using the system default method for displaying links.

5        LinkClickedEventArgs class (System.Windows.Forms)

6        a)        *ToString*

7  
8  
9 *Description*

10 Provides data for the **System.Windows.Forms.RichTextBox.LinkClicked** event.

11 A **System.Windows.Forms.LinkClickedEventArgs** specifies the text of the  
12 link that is clicked in the **System.Windows.Forms.RichTextBox** .

13        b)        *LinkClickedEventArgs*

14        *Example Syntax:*

15        c)        *ToString*

16  
17 [C#]        public        LinkClickedEventArgs(string        linkText);

18 [C++]        public:        LinkClickedEventArgs(String\*        linkText);

19 [VB]        Public        Sub        New(ByVal        linkText        As        String)

20 [JScript]        public        function        LinkClickedEventArgs(linkText        :        String);

21  
22 *Description*

23 Initializes a new instance of the  
24 **System.Windows.Forms.LinkClickedEventArgs** class.



The text specified in the *linkText* parameter can be a URL, file path, or other data that you want specified for the link being clicked. The text of the link that is clicked in the **System.Windows.Forms.RichTextBox** control.

d) *LinkText*

e) *ToString*

|           |         |            |          |                      |
|-----------|---------|------------|----------|----------------------|
| [C#]      | public  | string     | LinkText | {get;}               |
| [C++]     | public: | __property | String*  | get_LinkText();      |
| [VB]      | Public  | ReadOnly   | Property | LinkText As String   |
| [JScript] | public  | function   | get      | LinkText() : String; |

#### *Description*

Gets the text of the link being clicked.

The text returned by this property could be a file, URL, or other data specified by the link that is clicked.

LinkClickedEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **System.Windows.Forms.RichTextBox.LinkClicked** event of a **System.Windows.Forms.RichTextBox**. The source of the object. The **System.Windows.Forms.LinkClickedEventArgs** that contains the event data.

When you create a **System.Windows.Forms.LinkClickedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about delegates, see .

LinkLabel.LinkCollection class (System.Windows.Forms)

*a) ToString*

*Description*

Represents the collection of links within a **System.Windows.Forms.LinkLabel** control.

The **System.Windows.Forms.LinkLabel.LinkCollection** class stores the link displayed in the **System.Windows.Forms.LinkLabel** control. Each item in the collection is an instance of the **System.Windows.Forms.LinkLabel.Link** class, which defines the information of the link.

*b) LinkLabel.LinkCollection*

*Example Syntax:*

*c) ToString*

```
[C#]      public      LinkLabel.LinkCollection(LinkLabel      owner);
[C++]      public:      LinkCollection(LinkLabel*      owner);
[VB]      Public      Sub      New(ByVal      owner      As      LinkLabel)
[JScript] public function LinkLabel.LinkCollection(owner : LinkLabel);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.LinkLabel.LinkCollection** class.

An instance of this class cannot be created without associating it with a **System.Windows.Forms.LinkLabel** control. The **System.Windows.Forms.LinkLabel** control that owns the collection.

d) *Count*

e) *ToString*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | int        | Count    | {get;}           |
| [C++]     | public: | __property | int      | get_Count();     |
| [VB]      | Public  | ReadOnly   | Property | Count As Integer |
| [JScript] | public  | function   | get      | Count() : int;   |

*Description*

Gets the number of links in the collection.

This property enables you to determine the number of links in the **System.Windows.Forms.LinkLabel** control. You can then use this value when looping through the values of the collection and need to provide a number of iterations to perform the loop.

f) *IsReadOnly*

g) *ToString*

|           |         |            |            |                         |
|-----------|---------|------------|------------|-------------------------|
| [C#]      | public  | bool       | IsReadOnly | {get;}                  |
| [C++]     | public: | __property | bool       | get_IsReadOnly();       |
| [VB]      | Public  | ReadOnly   | Property   | IsReadOnly As Boolean   |
| [JScript] | public  | function   | get        | IsReadOnly() : Boolean; |

*Description*

Gets a value indicating whether this collection is read-only.

This property is always **false** for this collection.

h) *Item*

i) *ToString*

[C#] public virtual LinkLabel.Link this[int index] {get; set;}

[C++] public: \_\_property virtual LinkLabel.Link\* get\_Item(int index);public:  
\_\_property virtual void set\_Item(int index, LinkLabel.Link\*);

[VB] Overridable Public Default Property Item(ByVal index As Integer) As  
LinkLabel.Link

[JScript] return Value =

LinkCollectionObject.Item(index);LinkCollectionObject.Item(index) =

return Value;

*Description*

Gets and sets the link at the specified index within the collection.

You can use this method to obtain the link stored at a specific location within the collection. To determine the index of a specific item within the collection, use the **System.Windows.Forms.LinkLabel.LinkCollection.IndexOf(System.Windows.Forms.LinkLabel.Link)** method. The index of the link in the collection to get.

j) *Add*

[C#] public Link Add(int start, int length);

[C++] public: Link\* Add(int start, int length);

[VB] Public Function Add(ByVal start As Integer, ByVal length As Integer) As  
Link

[JScript] public function Add(start : int, length : int) : Link; Adds a link to the  
collection.

## Description

Adds a link to the collection.

*Return Value:* A **System.Windows.Forms.LinkLabel.Link** object representing the link that was created and added to the collection.

A **System.Windows.Forms.LinkLabel** control can display multiple links within the text of the control. The

**System.Windows.Forms.LinkLabel.LinkCollection.Add(System.Int32, System.Int32)** method enables you to convert text within the

**System.Windows.Forms.LinkLabel** control to a link that can be clicked on by the user to perform tasks similar to a **System.Windows.Forms.Button** control. This method adds the link that is created to the

**System.Windows.Forms.LinkLabel.LinkCollection** for the

**System.Windows.Forms.LinkLabel**. For example, if you want to set the word "quick" in the label text, "The quick brown fox", you call this method with the *start* parameter set to the value of four (4), and the *length* parameter to five (5). The word "quick" then changes to a link and the link is added to the collection. If you want to associate information with the link, such as the URL to display or a file to open when the user clicks on the link, use the other version of the

**System.Windows.Forms.LinkLabel.LinkCollection.Add(System.Int32, System.Int32)** method. The starting character within the text of the label where the link is created. The number of characters after the starting character to include in the link text.

### k) Add

```
[C#] public Link Add(int start, int length, object linkData);
```

```
[C++] public: Link* Add(int start, int length, Object* linkData);
```

```
[VB] Public Function Add(ByVal start As Integer, ByVal length As Integer,  
ByVal linkData As Object) As Link
```

```
[JScript] public function Add(start : int, length : int, linkData : Object) : Link;
```

## Description

Adds a link to the collection with information to associate with the link.

*Return Value:* A **System.Windows.Forms.LinkLabel.Link** object representing the link that was created and added to the collection.

A **System.Windows.Forms.LinkLabel** control can display multiple links within the text of the control. The

**System.Windows.Forms.LinkLabel.LinkCollection.Add(System.Int32, System.Int32)** method enables you to convert text within the

**System.Windows.Forms.LinkLabel** control to a link that can be clicked on by the user to perform tasks similar to a **System.Windows.Forms.Button**

control. This method adds the link that is created to the

**System.Windows.Forms.LinkLabel.LinkCollection** for the

**System.Windows.Forms.LinkLabel**. For example, if you want to set the word "quick" in the label text, "The quick brown fox", you call this method with the *start* parameter set to the value of four (4), and the *length* parameter to five (5). The word "quick" then changes to a link and the link is added to the

collection. This version of the Add method enables you to provide additional information that can be associated with the link through the *linkData* parameter.

For example, you can pass a **System.String** object to the *linkData* parameter that contains a URL to display when the link is clicked. You can then use this information in your handler for the

**System.Windows.Forms.LinkLabel.LinkClicked** event of the

**System.Windows.Forms.LinkLabel** control to display the URL in Microsoft Internet Explorer. The starting character within the text of the label where the link is created. The number of characters after the starting character to include in the link text. The object containing the information to associate with the link.

#### *l) Clear*

|           |             |          |      |          |
|-----------|-------------|----------|------|----------|
| [C#]      | public      | virtual  | void | Clear(); |
| [C++]     | public:     | virtual  | void | Clear(); |
| [VB]      | Overridable | Public   | Sub  | Clear()  |
| [JScript] | public      | function |      | Clear(); |

#### *Description*

Clears all links from the collection.

When you remove links from the collection, all information about the removed links is deleted. To remove a single link from the

**System.Windows.Forms.LinkLabel** , use the **System.Windows.Forms.LinkLabel.LinkCollection.Remove(System.Windows.Forms.LinkLabel.Link)** or **System.Windows.Forms.LinkLabel.LinkCollection.RemoveAt(System.Int32)** method.

*m) Contains*

```
[C#]          public          bool          Contains(LinkLabel.Link          link);
[C++]          public:          bool          Contains(LinkLabel.Link*          link);
[VB] Public Function Contains(ByVal link As LinkLabel.Link) As Boolean
[JavaScript] public function Contains(link : LinkLabel.Link) : Boolean;
```

*Description*

Determines whether the specified link is within the collection.

*Return Value:* **true** if the specified link is within the collection; otherwise, **false** .

The **System.Windows.Forms.LinkLabel.LinkCollection.Contains(System.Windows.Forms.LinkLabel.Link)** method enables you to determine whether a **System.Windows.Forms.LinkLabel.Link** object is a member of the collection. Once you know that the link is located within the collection, you can use the **System.Windows.Forms.LinkLabel.LinkCollection.IndexOf(System.Windows.Forms.LinkLabel.Link)** method to determine where the link is located within the collection. A **System.Windows.Forms.LinkLabel.Link** representing the link to search for in the collection.

*n) GetEnumerator*

```
[C#]          public          IEnumerator          GetEnumerator();
[C++]          public:          __sealed          IEnumerator*          GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JavaScript] public function GetEnumerator() : IEnumerator;
```

## Description

Returns an enumerator to use to iterate through the link collection.

**Return Value:** An **System.Collections.IEnumerator** object that represents the link collection.

### *o) IndexOf*

[C#] public int IndexOf(LinkLabel.Link link);

[C++] public: int IndexOf(LinkLabel.Link\* link);

[VB] Public Function IndexOf(ByVal link As LinkLabel.Link) As Integer

[JScript] public function IndexOf(link : LinkLabel.Link) : int;

## Description

Returns the index of the specified link within the collection.

**Return Value:** The zero-based index where the link is located within the collection; otherwise, negative one (-1).

The **System.Windows.Forms.LinkLabel.LinkCollection.IndexOf(System.Windows.Forms.LinkLabel.Link)** method enables you to determine where a link is located within the collection. To determine if a link is located within the collection before calling this method, use the **System.Windows.Forms.LinkLabel.LinkCollection.Contains(System.Windows.Forms.LinkLabel.Link)** method. A **System.Windows.Forms.LinkLabel.Link** representing the link to search for in the collection.

### *p) Remove*

[C#] public void Remove(LinkLabel.Link value);

[C++] public: void Remove(LinkLabel.Link\* value);

[VB] Public Sub Remove(ByVal value As LinkLabel.Link)



[JScript] public function Remove(value : LinkLabel.Link);

### Description

Removes the specified link from the collection.

When you remove a link from the collection, the indices change for subsequent items in the collection. All information about the removed item is deleted. You can use this method to remove a specific link from the collection by specifying the actual item to remove from the collection. To specify the index of the link to remove instead of the link itself, use the **System.Windows.Forms.LinkLabel.LinkCollection.RemoveAt(System.Int32)** method. To remove all links from the collection, use the **System.Windows.Forms.LinkLabel.LinkCollection.Clear** method. A **System.Windows.Forms.LinkLabel.Link** that represents the link to remove from the collection.

### q) RemoveAt

[C#] public void RemoveAt(int index);

[C++] public: \_\_sealed void RemoveAt(int index);

[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int);

### Description

Removes a link at a specified location within the collection.

When you remove a link from the collection, the indexes change for subsequent items in the collection. All information about the deleted link is lost. You can use this method to remove a specific link from the collection by specifying the index of the link to remove from the collection. To specify the link to remove instead of the index to the link, use the **System.Windows.Forms.LinkLabel.LinkCollection.Remove(System.Windows.Forms.LinkLabel.Link)** method. To remove all links from the collection, use the **System.Windows.Forms.LinkLabel.LinkCollection.Clear** method. The zero-based index of the item to remove from the collection.

**r) *ICollection.CopyTo***

[C#] void ICollection.CopyTo(Array dest, int index);

[C++] void ICollection::CopyTo(Array\* dest, int index);

[VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements  
ICollection.CopyTo

[JScript] function ICollection.CopyTo(dest : Array, index : int);

**s) *IList.Add***

[C#] int IList.Add(object value);

[C++] int IList::Add(Object\* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

**t) *IList.Contains***

[C#] bool IList.Contains(object link);

[C++] bool IList::Contains(Object\* link);

[VB] Function Contains(ByVal link As Object) As Boolean Implements  
IList.Contains

[JScript] function IList.Contains(link : Object) : Boolean;

**u) *IList.IndexOf***

[C#] int IList.IndexOf(object link);

[C++] int IList::IndexOf(Object\* link);

[VB] Function IndexOf(ByVal link As Object) As Integer Implements

1 IList.IndexOf

2 [JScript] function IList.IndexOf(link : Object) : int;

3 v) *IList.Insert*

4  
5 [C#] void IList.Insert(int index, object value);

6 [C++] void IList::Insert(int index, Object\* value);

7 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

8 IList.Insert

9 [JScript] function IList.Insert(index : int, value : Object);

10 w) *IList.Remove*

11  
12 [C#] void IList.Remove(object value);

13 [C++] void IList::Remove(Object\* value);

14 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

15 [JScript] function IList.Remove(value : Object);

16 LinkLabel class (System.Windows.Forms)

17 a) *ToString*

18  
19  
20 *Description*

21 Represents a Windows label control that can display hyperlinks.

22 The **System.Windows.Forms.LinkLabel** control is similar to a  
23 **System.Windows.Forms.Label** control with the exception that it can display a  
24 hyperlink. Multiple hyperlinks can be specified in the text of the control. Each  
25 hyperlink can perform a different task within an application. For example, you  
can use a hyperlink to display a Web site in Microsoft Internet Explorer or to load  
a log file associated with an application.

**b) *LinkLabel***

*Example Syntax:*

**c) *ToString***

|           |                 |              |
|-----------|-----------------|--------------|
| [C#]      | public          | LinkLabel(); |
| [C++]     | public:         | LinkLabel(); |
| [VB]      | Public Sub      | New()        |
| [JScript] | public function | LinkLabel(); |

*Description*

Initializes a new default instance of the **System.Windows.Forms.LinkLabel** class.

**d) *AccessibilityObject***

**e) *AccessibleDefaultActionDescription***

**f) *AccessibleDescription***

**g) *AccessibleName***

**h) *AccessibleRole***

**i) *ActiveLinkColor***

**j) *ToString***

*Description*

Gets or sets the color used to display an active link.

An active link is a link that is in the process of being clicked. This is similar to the depressed state of a **System.Windows.Forms.Button** control. You can use

this property to specify the color that the link is displayed in when the link is in the process of being clicked.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

MS1-861US.APP

|    |            |                                |
|----|------------|--------------------------------|
| 1  | <b>k)</b>  | <b><i>AllowDrop</i></b>        |
| 2  | <b>l)</b>  | <b><i>Anchor</i></b>           |
| 3  | <b>m)</b>  | <b><i>AutoSize</i></b>         |
| 4  | <b>n)</b>  | <b><i>BackColor</i></b>        |
| 5  | <b>o)</b>  | <b><i>BackgroundImage</i></b>  |
| 6  | <b>p)</b>  | <b><i>BindingContext</i></b>   |
| 7  | <b>q)</b>  | <b><i>BorderStyle</i></b>      |
| 8  | <b>r)</b>  | <b><i>Bottom</i></b>           |
| 9  | <b>s)</b>  | <b><i>Bounds</i></b>           |
| 10 | <b>t)</b>  | <b><i>CanFocus</i></b>         |
| 11 | <b>u)</b>  | <b><i>CanSelect</i></b>        |
| 12 | <b>v)</b>  | <b><i>Capture</i></b>          |
| 13 | <b>w)</b>  | <b><i>CausesValidation</i></b> |
| 14 | <b>x)</b>  | <b><i>ClientRectangle</i></b>  |
| 15 | <b>y)</b>  | <b><i>ClientSize</i></b>       |
| 16 | <b>z)</b>  | <b><i>CompanyName</i></b>      |
| 17 | <b>aa)</b> | <b><i>Container</i></b>        |
| 18 | <b>bb)</b> | <b><i>ContainsFocus</i></b>    |
| 19 | <b>cc)</b> | <b><i>ContextMenu</i></b>      |
| 20 | <b>dd)</b> | <b><i>Controls</i></b>         |
| 21 | <b>ee)</b> | <b><i>Created</i></b>          |
| 22 | <b>ff)</b> | <b><i>CreateParams</i></b>     |
| 23 | <b>gg)</b> | <b><i>Cursor</i></b>           |
| 24 |            |                                |
| 25 |            |                                |

1        *hh) DataBindings*

2        *ii) DefaultImeMode*

3        *jj) DefaultSize*

4        *kk) DesignMode*

5        *ll) DisabledLinkColor*

6        *mm) ToString*

7  
8  
9        *Description*

10       Gets or sets the color used when displaying a disabled link.

11       This property enables you to specify the color for links that are disabled in the  
12       **System.Windows.Forms.LinkLabel** . Disabled links do not cause the  
13       **System.Windows.Forms.LinkLabel.LinkClicked** event to be raised.

MSI-861US.APP  
509-324-9256  
lee@hayes.plc

|    |                                    |
|----|------------------------------------|
| 1  | <b><i>nn) DisplayRectangle</i></b> |
| 2  | <b><i>oo) Disposing</i></b>        |
| 3  | <b><i>pp) Dock</i></b>             |
| 4  | <b><i>qq) Enabled</i></b>          |
| 5  | <b><i>rr) Events</i></b>           |
| 6  | <b><i>ss) FlatStyle</i></b>        |
| 7  | <b><i>tt) Focused</i></b>          |
| 8  | <b><i>uu) Font</i></b>             |
| 9  | <b><i>vv) FontHeight</i></b>       |
| 10 | <b><i>ww) ForeColor</i></b>        |
| 11 | <b><i>xx) Handle</i></b>           |
| 12 | <b><i>yy) HasChildren</i></b>      |
| 13 | <b><i>zz) Height</i></b>           |
| 14 | <b><i>aaa) Image</i></b>           |
| 15 | <b><i>bbb) ImageAlign</i></b>      |
| 16 | <b><i>ccc) ImageIndex</i></b>      |
| 17 | <b><i>ddd) ImageList</i></b>       |
| 18 | <b><i>eee) ImeMode</i></b>         |
| 19 | <b><i>fff) InvokeRequired</i></b>  |
| 20 | <b><i>ggg) IsAccessible</i></b>    |
| 21 | <b><i>hhh) IsDisposed</i></b>      |
| 22 | <b><i>iii) IsHandleCreated</i></b> |
| 23 | <b><i>jjj) Left</i></b>            |
| 24 |                                    |
| 25 |                                    |





This property enables you to specify the behavior of links when they are displayed in the control. For example, if you want links to be displayed with an underline only when the mouse pointer is over a link, you can set this property to **LinkBehavior.HoverUnderline** . For more information on the types of behaviors that can be associated with a link, see **System.Windows.Forms.LinkBehavior** .

*ooo) LinkColor*

*ppp) ToString*

[C#]            public            Color            LinkColor            {get;            set;}

[C++]   public:   \_\_property   Color   get\_LinkColor();public:   \_\_property   void  
set\_LinkColor(Color);

[VB]            Public            Property            LinkColor            As            Color

[JScript]   public   function   get   LinkColor()   :   Color;public   function   set  
LinkColor(Color);

### *Description*

Gets or sets the color used when displaying a normal link.

This property enables you to specify the color that is initially displayed for all links in the **System.Windows.Forms.LinkLabel** .

*qqq) Links*

*rrr) ToString*

[C#]            public            LinkLabel.LinkCollection            Links            {get;}

[C++]   public:   \_\_property   LinkLabel.LinkCollection\*   get\_Links();

[VB]   Public   ReadOnly   Property   Links   As   LinkLabel.LinkCollection

[JScript]   public   function   get   Links()   :   LinkLabel.LinkCollection;

## Description

Gets the collection of links contained within the **System.Windows.Forms.LinkLabel** .

A **System.Windows.Forms.LinkLabel** control can display any number of links within the text of the control. This property enables you to access the **System.Windows.Forms.LinkLabel.LinkCollection** instance associated with the **System.Windows.Forms.LinkLabel** that represents the collection of links displayed in the control. You can then use the members of the **System.Windows.Forms.LinkLabel.LinkCollection** class to add, remove, and find links in the collection. For more information on the types of tasks you can perform with the link collection, see **System.Windows.Forms.LinkLabel.LinkCollection** .

*sss) LinkVisited*

*ttt) ToString*

```
[C#]          public          bool          LinkVisited          {get;          set;}
```

```
[C++] public: __property bool get_LinkVisited();public: __property void  
set_LinkVisited(bool);
```

```
[VB]          Public          Property          LinkVisited          As          Boolean
```

```
[JScript] public function get LinkVisited() : Boolean;public function set  
LinkVisited(Boolean);
```

## Description

Gets or sets a value indicating whether a link should be displayed as though it were visited.

A **System.Windows.Forms.LinkLabel** control does not automatically denote that a link is a visited link. To display the link as a visited link, you can set the value of this property to **true** in an event handler for the **System.Windows.Forms.LinkLabel.LinkClicked** event of a **System.Windows.Forms.LinkLabel** . A visited link is displayed using the color specified in the **System.Windows.Forms.LinkLabel.VisitedLinkColor**

property of the **System.Windows.Forms.LinkLabel** control. Once the form containing the **System.Windows.Forms.LinkLabel** control is closed, the all display state associated with the link is deleted. In order to retain the display state of the link, you need to store the display state of the link in a registry setting associated with your application.

*uuu) Location*

*vvv) Name*

*www) OverrideCursor*

*xxx) ToString*

*Description*

yyy) *Parent*  
 zzz) *PreferredHeight*  
 aaaa) *PreferredWidth*  
 bbbb) *ProductName*  
 cccc) *ProductVersion*  
 dddd) *RecreatingHandle*  
 eeee) *Region*  
 ffff) *RenderRightToLeft*  
 gggg) *RenderTransparent*  
 hhhh) *ResizeRedraw*  
 iii) *Right*  
 jjjj) *RightToLeft*  
 kkkk) *ShowFocusCues*  
 llll) *ShowKeyboardCues*  
 mmmm) *Site*  
 nnnn) *Size*  
 oooo) *TabIndex*  
 pppp) *TabStop*  
 qqqq) *Tag*  
 rrrr) *Text*  
 ssss) *ToString*

*Description*

*tttt) TextAlign*

*uuuu) Top*

*vvvv) TopLevelControl*

*www)UseMnemonic*

*xxxx) Visible*

*yyyy) VisitedLinkColor*

*zzzz) ToString*

*Description*

Gets or sets the color used when displaying a link that that has been previously visited.

This property enables you to specify the color that is displayed for all links in the **LinkLabelSystem.Windows.Forms** that have been visited by the user.

*aaaaa) Width*

*bbbbb) WindowTarget*

*cccc) ToString*

*Description*

Occurs when a link is clicked within the control.

Typically, the **System.Windows.Forms.LinkLabel.LinkClicked** event is handled to perform tasks when the user clicks on a link in the control. The event handler for the **System.Windows.Forms.LinkLabel.LinkClicked** event is passed an instance of the **System.Windows.Forms.LinkLabelLinkClickedEventArgs** class that contains a **System.Windows.Forms.LinkLabel.Link** object that is associated with the link that was clicked. You can use information specified in the **System.Windows.Forms.LinkLabel.Link.LinkData** property of **System.Windows.Forms.LinkLabel.Link** class to determine which link was clicked or what type of task to perform when the link is clicked. For example, if a **System.Windows.Forms.LinkLabel** control has a **System.Windows.Forms.LinkLabel.Link** object defined with its **System.Windows.Forms.LinkLabel.Link.LinkData** property set to the string `www.microsoft.com`, you can use this information in an event handler for the **System.Windows.Forms.LinkLabel.LinkClicked** event to display the Web site.

#### *dddd) CreateAccessibilityInstance*

```
[C#]    protected    override    AccessibleObject    CreateAccessibilityInstance();
[C++]    protected:    AccessibleObject*    CreateAccessibilityInstance();
[VB]    Overrides    Protected    Function    CreateAccessibilityInstance()    As
AccessibleObject
[JScript]    protected    override    function    CreateAccessibilityInstance()    :
AccessibleObject;
```

#### *Description*

Constructs the new instance of the accessibility object for this control. Subclasses should not call `base.CreateAccessibilityObject`.

#### *eeee) CreateHandle*

```
[C#]    protected    override    void    CreateHandle();
[C++]    protected:    void    CreateHandle();
```

|           |           |           |          |                 |
|-----------|-----------|-----------|----------|-----------------|
| [VB]      | Overrides | Protected | Sub      | CreateHandle()  |
| [JScript] | protected | override  | function | CreateHandle(); |

*Description*

Creates a handle for this control. This method is called by the .NET Framework, this should not be called. Inheriting classes should always call base.createHandle when overriding this method.

**fffff) OnEnabledChanged**

|           |            |           |          |                                        |
|-----------|------------|-----------|----------|----------------------------------------|
| [C#]      | protected  | override  | void     | OnEnabledChanged(EventArgs e);         |
| [C++]     | protected: |           | void     | OnEnabledChanged(EventArgs* e);        |
| [VB]      | Overrides  | Protected | Sub      | OnEnabledChanged(ByVal e As EventArgs) |
| [JScript] | protected  | override  | function | OnEnabledChanged(e : EventArgs);       |

*Description*

**ggggg) OnFontChanged**

|           |            |           |          |                                     |
|-----------|------------|-----------|----------|-------------------------------------|
| [C#]      | protected  | override  | void     | OnFontChanged(EventArgs e);         |
| [C++]     | protected: |           | void     | OnFontChanged(EventArgs* e);        |
| [VB]      | Overrides  | Protected | Sub      | OnFontChanged(ByVal e As EventArgs) |
| [JScript] | protected  | override  | function | OnFontChanged(e : EventArgs);       |

*Description*



#### *hhhhh)OnGotFocus*

```
[C#]    protected    override    void    OnGotFocus(EventArgs    e);
[C++]    protected:    void    OnGotFocus(EventArgs*    e);
[VB]    Overrides    Protected    Sub    OnGotFocus(ByVal e As EventArgs)
[JScript]    protected    override    function    OnGotFocus(e : EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.GotFocus** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *iiii) OnKeyDown*

```
[C#]    protected    override    void    OnKeyDown(KeyEventArgs    e);
[C++]    protected:    void    OnKeyDown(KeyEventArgs*    e);
[VB]    Overrides    Protected    Sub    OnKeyDown(ByVal e As KeyEventArgs)
[JScript]    protected    override    function    OnKeyDown(e : KeyEventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.OnKeyDown(System.Windows.Forms.KeyEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.KeyEventArgs** that contains the event data.

#### *jjjjj) OnLinkClicked*

[C#] protected virtual void OnLinkClicked(LinkLabelLinkClickedEventArgs e);

[C++] protected: virtual void OnLinkClicked(LinkLabelLinkClickedEventArgs\* e);

[VB] Overridable Protected Sub OnLinkClicked(ByVal e As LinkLabelLinkClickedEventArgs)

[JScript] protected function OnLinkClicked(e : LinkLabelLinkClickedEventArgs);

#### *Description*

Raises the **System.Windows.Forms.LinkLabel.LinkClicked** event.

Raising an event invokes the event handler through a delegate. For more information, see . A

**System.Windows.Forms.LinkLabelLinkClickedEventArgs** that contains the event data.

#### *kkkkk) OnLostFocus*

[C#] protected override void OnLostFocus(EventArgs e);

[C++] protected: void OnLostFocus(EventArgs\* e);

[VB] Overrides Protected Sub OnLostFocus(ByVal e As EventArgs)

[JScript] protected override function OnLostFocus(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.LostFocus** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### IIII) *OnMouseDown*

```
[C#]    protected    override    void    OnMouseDown(MouseEventArgs    e);
[C++]    protected:    void    OnMouseDown(MouseEventArgs*    e);
[VB] Overrides Protected Sub OnMouseDown(ByVal e As MouseEventArgs)
[JScript] protected override function OnMouseDown(e : MouseEventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.OnMouseDown(System.Windows.Forms.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

### mmmm) *OnMouseLeave*

```
[C#]    protected    override    void    OnMouseLeave(EventArgs    e);
[C++]    protected:    void    OnMouseLeave(EventArgs*    e);
[VB] Overrides Protected Sub OnMouseLeave(ByVal e As EventArgs)
[JScript] protected override function OnMouseLeave(e : EventArgs);
```

#### *Description*

Raises the **System.Windows.Forms.Control.OnMouseLeave(System.EventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

## *nnnnn)OnMouseMove*

```
1
2 [C#]    protected    override    void    OnMouseMove(MouseEventArgs    e);
3
4 [C++]    protected:    void    OnMouseMove(MouseEventArgs*    e);
5
6 [VB] Overrides Protected Sub OnMouseMove(ByVal e As MouseEventArgs)
7
8 [JScript] protected override function OnMouseMove(e : MouseEventArgs);
9
```

### *Description*

Raises the **System.Windows.Forms.Control.OnMouseMove(System.Windows.Forms.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

## *ooooo) OnMouseUp*

```
13
14
15 [C#]    protected    override    void    OnMouseUp(MouseEventArgs    e);
16
17 [C++]    protected:    void    OnMouseUp(MouseEventArgs*    e);
18
19 [VB] Overrides Protected Sub OnMouseUp(ByVal e As MouseEventArgs)
20
21 [JScript] protected override function OnMouseUp(e : MouseEventArgs);
22
```

### *Description*

Raises the **System.Windows.Forms.Control.OnMouseUp(System.Windows.Forms.MouseEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MouseEventArgs** that contains the event data.

### *ppppp) OnPaint*

```
[C#]    protected    override    void    OnPaint(PaintEventArgs    e);
[C++]    protected:    void    OnPaint(PaintEventArgs*    e);
[VB]    Overrides    Protected    Sub    OnPaint(ByVal e As PaintEventArgs)
[JScript]    protected    override    function    OnPaint(e : PaintEventArgs);
```

#### *Description*

Raises the

**System.Windows.Forms.Control.OnPaint(System.Windows.Forms.PaintEventArgs)** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.PaintEventArgs** that contains the event data.

### *qqqqq) OnPaintBackground*

```
[C#]    protected    override    void    OnPaintBackground(PaintEventArgs    e);
[C++]    protected:    void    OnPaintBackground(PaintEventArgs*    e);
[VB]    Overrides    Protected    Sub    OnPaintBackground(ByVal e As PaintEventArgs)
[JScript]    protected    override    function    OnPaintBackground(e : PaintEventArgs);
```

#### *Description*

### *rrrrr) OnTextAlignChanged*

```
[C#]    protected    override    void    OnTextAlignChanged(EventArgs    e);
[C++]    protected:    void    OnTextAlignChanged(EventArgs*    e);
```

1 [VB] Overrides Protected Sub OnTextAlignChanged(ByVal e As EventArgs)

2 [JScript] protected override function OnTextAlignChanged(e : EventArgs);

4 *Description*

6 *sssss) OnTextChanged*

8 [C#] protected override void OnTextChanged(EventArgs e);

9 [C++] protected: void OnTextChanged(EventArgs\* e);

10 [VB] Overrides Protected Sub OnTextChanged(ByVal e As EventArgs)

11 [JScript] protected override function OnTextChanged(e : EventArgs);

13 *Description*

15 *tttt) PointInLink*

17 [C#] protected Link PointInLink(int x, int y);

18 [C++] protected: Link\* PointInLink(int x, int y);

19 [VB] Protected Function PointInLink(ByVal x As Integer, ByVal y As Integer) As

20 Link

21 [JScript] protected function PointInLink(x : int, y : int) : Link;

23 *Description*

24 Gets the link located at the specified client coordinates.

25 *Return Value:* A **System.Windows.Forms.LinkLabel.Link** representing the

link located at the specified coordinates. If the point does not contain a link, **null** is returned.

This method enables you to determine whether a link is located at a specific point within a **System.Windows.Forms.LinkLabel** control. You can use this method in an event handler for the **System.Windows.Forms.Control.MouseEnter** event of the control to determine whether the mouse pointer is hovering over a link in the control. Once you have determined that the mouse pointer is over a link, you can then display additional information about the link to the user through **System.Windows.Forms.StatusBar** text or a **System.Windows.Forms.ToolTip**. The horizontal coordinate of the point to search for a link. The vertical coordinate of the point to search for a link.

*uuuuu)ProcessDialogKey*

[C#]      protected      override      bool      ProcessDialogKey(Keys      keyData);

[C++]      protected:      bool      ProcessDialogKey(Keys      keyData);

[VB] Overrides Protected Function ProcessDialogKey(ByVal keyData As Keys)

As      Boolean

[JScript] protected override function ProcessDialogKey(keyData : Keys) :

Boolean;

### *Description*

Processes a dialog key. This method is called during message pre-processing to handle dialog characters, such as TAB, RETURN, ESCAPE, and arrow keys. This method is called only if the `isInputKey()` method indicates that the control isn't interested in the key. `processDialogKey()` simply sends the character to the parent's `processDialogKey()` method, or returns false if the control has no parent. The `Form` class overrides this method to perform actual processing of dialog keys. When overriding `processDialogKey()`, a control should return true to indicate that it has processed the key. For keys that aren't processed by the control, the result of "`base.processDialogChar()`" should be returned. Controls will seldom, if ever, need to override this method.

*Return Value:* true to consume the key, false to allow further processing. key code and modifier flags.

### *vvvvv) Select*

[C#] protected override void Select(bool directed, bool forward);  
[C++] protected: void Select(bool directed, bool forward);  
[VB] Overrides Protected Sub Select(ByVal directed As Boolean, ByVal forward  
As Boolean)  
[JScript] protected override function Select(directed : Boolean, forward :  
Boolean);

### *Description*

### *wwwww)SetBoundsCore*

[C#] protected override void SetBoundsCore(int x, int y, int width, int height,  
BoundsSpecified specified);  
[C++] protected: void SetBoundsCore(int x, int y, int width, int height,  
BoundsSpecified specified);  
[VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As  
Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As  
BoundsSpecified)  
[JScript] protected override function SetBoundsCore(x : int, y : int, width : int,  
height : int, specified : BoundsSpecified);

### *Description*

Performs the work of setting the bounds of this control. Inheriting classes can  
override this function to add size restrictions. Inheriting classes must call



base.setBoundsCore to actually cause the bounds of the control to change. new left of the control. new right of the control. new width of the control. new height of the control. Which values were specified. This parameter is meant to reflect user intent, not indicate which values have changed.

*xxxxxx) IButtonControl.NotifyDefault*

```
[C#]          void          IButtonControl.NotifyDefault(bool          value);
[C++]          void          IButtonControl::NotifyDefault(bool          value);
[VB]  Sub      NotifyDefault(ByVal  value  As  Boolean)  Implements
IButtonControl.NotifyDefault
[JScript] function IButtonControl.NotifyDefault(value : Boolean);
```

*yyyyyy) IButtonControl.PerformClick*

```
[C#]          void          IButtonControl.PerformClick();
[C++]          void          IButtonControl::PerformClick();
[VB]  Sub      PerformClick()  Implements  IButtonControl.PerformClick
[JScript] function IButtonControl.PerformClick();
```

*zzzzzz) WndProc*

```
[C#]  protected  override  void  WndProc(ref  Message  msg);
[C++]          protected:  void  WndProc(Message*  msg);
[VB]  Overrides  Protected  Sub  WndProc(ByRef  msg  As  Message)
[JScript]  protected  override  function  WndProc(msg  :  Message);
```

*Description*

LinkLabelLinkClickedEventArgs class (System.Windows.Forms)

a) *WndProc*

*Description*

Provides data for the **System.Windows.Forms.LinkLabel.OnLinkClicked(System.Windows.Forms.LinkLabelLinkClickedEventArgs)** event.

b) *LinkLabelLinkClickedEventArgs*

*Example Syntax:*

c) *WndProc*

[C#] public LinkLabelLinkClickedEventArgs(LinkLabel.Link link);

[C++] public: LinkLabelLinkClickedEventArgs(LinkLabel.Link\* link);

[VB] Public Sub New(ByVal link As LinkLabel.Link)

[JScript] public function LinkLabelLinkClickedEventArgs(link : LinkLabel.Link);

*Description*

Initializes a new instance of the **System.Windows.Forms.LinkLabelLinkClickedEventArgs** class, given the link. The **System.Windows.Forms.LinkLabel.Link** that was clicked.

d) *Link*

e) *WndProc*

[C#] public LinkLabel.Link Link {get;}

[C++] public: \_\_property LinkLabel.Link\* get\_Link();

1 [VB] Public ReadOnly Property Link As LinkLabel.Link  
2 [JScript] public function get Link() : LinkLabel.Link;  
3

4 *Description*

5 Gets the **System.Windows.Forms.LinkLabel.Link** that was clicked.

6 LinkLabelLinkClickedEventHandler delegate (System.Windows.Forms)

7 a) *ToString*

10 *Description*

11 Represents the method that will handle the  
12 **System.Windows.Forms.LinkLabel.LinkClicked** event of a  
13 **System.Windows.Forms.LinkLabel** . The source of the event. A  
14 **System.Windows.Forms.LinkLabelLinkClickedEventArgs** that contains the  
15 event data.

16 When you create a  
17 **System.Windows.Forms.LinkLabelLinkClickedEventHandler** delegate,  
18 you identify the method that will handle the event. To associate the event with  
19 your event handler, add an instance of the delegate to the event. The event  
20 handler is called whenever the event occurs, until you remove the delegate. For  
21 more information about delegates, see .

18 LinkState enumeration (System.Windows.Forms)

19 a) *ToString*

20 b) *ToString*

23 *Description*

c) *ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | LinkState | Hover;       |
| [C++]     | public: | const | LinkState | Hover;       |
| [VB]      | Public  | Const | Hover     | As LinkState |
| [JScript] | public  | var   | Hover     | : LinkState; |

*Description*

d) *ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | LinkState | Normal;      |
| [C++]     | public: | const | LinkState | Normal;      |
| [VB]      | Public  | Const | Normal    | As LinkState |
| [JScript] | public  | var   | Normal    | : LinkState; |

*Description*

e) *ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | LinkState | Visited;     |
| [C++]     | public: | const | LinkState | Visited;     |
| [VB]      | Public  | Const | Visited   | As LinkState |
| [JScript] | public  | var   | Visited   | : LinkState; |

*Description*

ListBindingConverter class (System.Windows.Forms)

*a) ToString*

*Description*

Provides a type converter to convert **System.Windows.Forms.Binding** objects to and from various other representations.

The **System.Windows.Forms.ListBindingConverter** is used to evaluate and convert a property of an object into a **System.Windows.Forms.Binding**.

*b) ListBindingConverter*

*Example Syntax:*

*c) ToString*

|           |                 |                         |
|-----------|-----------------|-------------------------|
| [C#]      | public          | ListBindingConverter(); |
| [C++]     | public:         | ListBindingConverter(); |
| [VB]      | Public          | Sub New()               |
| [JScript] | public function | ListBindingConverter(); |

*d) CanConvertTo*

|       |                                                                                          |
|-------|------------------------------------------------------------------------------------------|
| [C#]  | public override bool CanConvertTo(ITypeDescriptorContext context, Type destinationType); |
| [C++] | public: bool CanConvertTo(ITypeDescriptorContext* context, Type*                         |

destinationType);

```
[VB] Overrides Public Function CanConvertTo(ByVal context As
ITypeDescriptorContext, ByVal destinationType As Type) As Boolean
[JScrip] public override function CanConvertTo(context :
ITypeDescriptorContext, destinationType : Type) : Boolean;
```

### *Description*

Gets a value indicating whether this converter can convert an object to the given destination type using the context.

*Return Value:* **true** if this converter can perform the conversion; otherwise, **false**.

The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null**, so always check. Also, properties on the context object can return **null**. An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you wish to convert to.

### *e) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
culture, Object* value, Type* destinationType);
[VB] Overrides Public Function ConvertTo(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
ByVal destinationType As Type) As Object
[JScrip] public override function ConvertTo(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object, destinationType : Type) : Object;
```

## Description

Converts the given object to another type. The most common types to convert are to and from a string object. The default implementation will make a call to ToString on the object if the object is valid and if the destination type is string. If this cannot convert to the destination type, this will throw a NotSupportedException.

**Return Value:** The converted object. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null. An optional culture info. If not supplied the current culture is assumed. The object to convert. The type to convert the object to.

### f) CreateInstance

```
[C#] public override object CreateInstance(ITypeDescriptorContext context,
IDictionary propertyValues);
[C++] public: Object* CreateInstance(ITypeDescriptorContext* context,
IDictionary* propertyValues);
[VB] Overrides Public Function CreateInstance(ByVal context As
ITypeDescriptorContext, ByVal propertyValues As IDictionary) As Object
[JScript] public override function CreateInstance(context :
ITypeDescriptorContext, propertyValues : IDictionary) : Object;
```

## Description

Creates an instance of this type given a set of property values for the object. This is useful for objects that are immutable, but still want to provide changable properties.

**Return Value:** The newly created object, or null if the object could not be created. The default implementation returns null. A type descriptor through which additional context may be provided. A dictionary of new property values.

The dictionary contains a series of name-value pairs, one for each property returned from `GetProperties`.

*g) GetCreateInstanceSupported*

[C#] public override bool GetCreateInstanceSupported(ITypeDescriptorContext context);

[C++] public: bool GetCreateInstanceSupported(ITypeDescriptorContext\* context);

[VB] Overrides Public Function GetCreateInstanceSupported(ByVal context As ITypeDescriptorContext) As Boolean

[JScript] public override function GetCreateInstanceSupported(context : ITypeDescriptorContext) : Boolean;

*Description*

Determines if changing a value on this object should require a call to `CreateInstance` to create a new value.

*Return Value:* Returns true if `CreateInstance` should be called when a change is made to one or more properties of this object. A type descriptor through which additional context may be provided.

ListBox class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a Windows list box control.

The **System.Windows.Forms.ListBox** control enables you to display a list of items to the user that the user can select by clicking. A

**System.Windows.Forms.ListBox** control can provide single or multiple selections using the **System.Windows.Forms.ListBox.SelectionMode**



property. The **System.Windows.Forms.ListBox** also provides the **System.Windows.Forms.ListBox.MultiColumn** property to enable the display of items in columns instead of a straight vertical list of items. This allows the control to display more visible items and prevents the need for the user to scroll to an item.

#### b) *ToString*

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C#]      | public  | const | int               | DefaultItemHeight; |
| [C++]     | public: | const | int               | DefaultItemHeight; |
| [VB]      | Public  | Const | DefaultItemHeight | As Integer         |
| [JScript] | public  | var   | DefaultItemHeight | : int;             |

#### *Description*

Specifies the default item height for an owner-drawn **System.Windows.Forms.ListBox**.

#### c) *ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | int       | NoMatches; |
| [C++]     | public: | const | int       | NoMatches; |
| [VB]      | Public  | Const | NoMatches | As Integer |
| [JScript] | public  | var   | NoMatches | : int;     |

#### *Description*

Specifies that no matches are found during a search.

This constant is returned by the **System.Windows.Forms.ListBox.FindString(System.String)**, **System.Windows.Forms.ListBox.FindStringExact(System.String)**, and **System.Windows.Forms.ListBox.IndexFromPoint(System.Drawing.Point)** methods when no matching values are found in a search.

**d) *ListBox***

*Example Syntax:*

e) *ToString*

|           |                 |            |
|-----------|-----------------|------------|
| [C#]      | public          | ListBox(); |
| [C++]     | public:         | ListBox(); |
| [VB]      | Public Sub      | New()      |
| [JScript] | public function | ListBox(); |

### Description

Initializes a new instance of the **System.Windows.Forms.ListBox** class.

### f) *AccessibilityObject*

**g) *AccessibleDefaultActionDescription***

### ***h) AccessibleDescription***

**i) *AccessibleName***

j) *AccessibleRole*

***k) AllowDrop***

**1) Anchor**

*m) BackColor*

***n) ToString***

### Description

o) *BackgroundImage*

p) *ToString*

[C#] public override Image BackgroundImage {get; set;}

[C++] public: \_\_property virtual Image\* get\_BackgroundImage();public:

\_\_property virtual void set\_BackgroundImage(Image\*);

[VB] Overrides Public Property BackgroundImage As Image

[JScript] public function get BackgroundImage() : Image;public function set

BackgroundImage(Image);

*Description*

q) *BindingContext*

r) *BorderStyle*

s) *ToString*

*Description*

Retrieves the current border style. Values for this are taken from The System.Windows.Forms.BorderStyle enumeration.

*Return Value:* the current border style.

- t) *Bottom*
- u) *Bounds*
- v) *CanFocus*
- w) *CanSelect*
- x) *Capture*
- y) *CausesValidation*
- z) *ClientRectangle*
- aa) *ClientSize*
- bb) *ColumnWidth*
- cc) *ToString*

### *Description*

Gets or sets the width of columns in a multicolumn **System.Windows.Forms.ListBox** .

If you set the value to zero (0), a default width is assigned to each column. If the **System.Windows.Forms.ListBox** is a multicolumn list box, this property returns the current width of each column in the list. You can use this property to ensure that each column in a multicolumn **System.Windows.Forms.ListBox** can properly display its items.

1        **dd)    *CompanyName***

2        **ee)    *Container***

3        **ff)    *ContainsFocus***

4        **gg)    *ContextMenu***

5        **hh)    *Controls***

6        **ii)    *Created***

7        **jj)    *CreateParams***

8        **kk)    *ToString***

11        *Description*

12        Retrieves the parameters needed to create the handle. Inheriting classes can  
13        override this to provide extra functionality. They should not, however, forget to  
14        call `base.CreateParams()` first to get the struct filled up with the basic info.

15        **ll)    *Cursor***

16        **mm)    *DataBindings***

17        **nn)    *DataManager***

18        **oo)    *DataSource***

19        **pp)    *DefaultImeMode***

20        **qq)    *DefaultSize***

21        **rr)    *ToString***

24        *Description*

*ss) DesignMode*  
*tt) DisplayMember*  
*uu) DisplayRectangle*  
*vv) Disposing*  
*ww) Dock*  
*xx) DrawMode*  
*yy) ToString*

## Description

Gets or sets the drawing mode for the control.

*zz) Enabled*  
*aaa) Events*  
*bbb) Focused*  
*ccc) Font*  
*ddd) FontHeight*  
*eee) ForeColor*  
*fff) ToString*

## Description

1        *ggg) Handle*

2        *hhh) HasChildren*

3        *iii) Height*

4        *jjj) HorizontalExtent*

5        *kkk) ToString*

6  
7  
8        *Description*

9        Gets or sets the width by which the horizontal scroll bar of a  
10        **System.Windows.Forms.ListBox** can scroll.

11        This property only reports a useful value if the  
12        **System.Windows.Forms.ListBox.HorizontalScrollbar** property is set to  
13        **true** . If the width of the **System.Windows.Forms.ListBox** is smaller than the  
14        value of this property, the horizontal scroll bar horizontally scrolls items in the  
15        **System.Windows.Forms.ListBox** . If the width of the  
16        **System.Windows.Forms.ListBox** is equal to or greater than this value, the  
17        horizontal scroll bar is hidden. The value of this property is not dynamically  
18        updated by the **System.Windows.Forms.ListBox** . This property is useful  
19        when the items of the **System.Windows.Forms.ListBox** are owner-drawn. For  
20        example, if the owner drawn items of the **System.Windows.Forms.ListBox**  
21        are 200 pixels wide, but the **System.Windows.Forms.ListBox** is 60 pixels  
22        wide, the **System.Windows.Forms.ListBox.HorizontalExtent** property  
23        would need to be set to 200 in order to scroll the right edge of the items into the  
24        visible region of the control.

19        *lll) HorizontalScrollbar*

20        *mmm) ToString*

22        [C#]        public        bool        HorizontalScrollbar        {get;        set;}

23        [C++] public: \_\_property bool get\_HorizontalScrollbar();public: \_\_property void  
24        set\_HorizontalScrollbar(bool);

25        [VB]        Public        Property        HorizontalScrollbar        As        Boolean

[JScript] public function get HorizontalScrollbar() : Boolean; public function set HorizontalScrollbar(Boolean);

#### *Description*

Gets or sets a value indicating whether a horizontal scroll bar is displayed in the control.

The **System.Windows.Forms.ListBox.HorizontalScrollbar** property determines whether the **System.Windows.Forms.ListBox** should display a horizontal scroll bar when the width of items within the **System.Windows.Forms.ListBox** extend beyond the right edge of the control. When this property is set to **true**, the scroll bar is automatically displayed based on the width of items in the **System.Windows.Forms.ListBox**. If the **System.Windows.Forms.ListBox** is an owner-drawn list box, in order to properly display a horizontal scroll bar, you must set the **System.Windows.Forms.ListBox.HorizontalExtent** property.

*nnn) ImeMode*

*ooo) IntegralHeight*

*ppp) ToString*

#### *Description*

Gets or sets a value indicating whether the control should resize to avoid showing partial items.

When this property is set to **true**, the control automatically resizes to ensure that an item is not partially displayed. If you want to maintain the original size of the **System.Windows.Forms.ListBox** based on the space requirements of your form, set this property to **false**. If the **System.Windows.Forms.ListBox** does not contain any items, this property has no effect.



1        *qqq) InvokeRequired*

2        *rrr) IsAccessible*

3        *sss) IsDisposed*

4        *ttt) IsHandleCreated*

5        *uuu) ItemHeight*

6        *vvv) ToString*

7  
8  
9        *Description*

10       Gets or sets the height of an item in the **System.Windows.Forms.ListBox** .

11       When the **System.Windows.Forms.ListBox.DrawMode** property is set to  
12       **DrawMode.OwnerDrawFixed** , all items have the same height. When the  
13       **System.Windows.Forms.ListBox.DrawMode** property is set to  
14       **DrawMode.OwnerDrawVariable** , the  
15       **System.Windows.Forms.ListBox.ItemHeight** property specifies the height  
16       of each item added to the **System.Windows.Forms.ListBox** . Because each  
17       item in an owner-drawn list can have a different height, you can use the  
18       **System.Windows.Forms.ListBox.GetItemHeight(System.Int32)** method  
19       to get the height of a specific item in the **System.Windows.Forms.ListBox** .  
20       If you use the **System.Windows.Forms.ListBox.ItemHeight** property on a  
21       **System.Windows.Forms.ListBox** with items of variable height, this property  
22       returns the height of the first item in the control.

18        *www) Items*

19        *xxx) ToString*

21       [C#]        public        ListBox.ObjectCollection        Items        {get;}

22       [C++]       public:       \_\_property       ListBox.ObjectCollection\*       get\_Items();

23       [VB]       Public       ReadOnly       Property       Items       As       ListBox.ObjectCollection

24       [JScript]   public       function       get       Items()       :       ListBox.ObjectCollection;

1  
2 *Description*

3 Gets the items of the **System.Windows.Forms.ListBox** .

4 This property enables you to obtain a reference to the list of items that are  
5 currently stored in the **System.Windows.Forms.ListBox** . With this reference,  
6 you can add items, remove items, and obtain a count of the items in the  
7 collection. For more information on the tasks that can be performed with the  
8 item collection, see the **System.Windows.Forms.ListBox.ObjectCollection**  
9 class reference topics.

10 *yyy) Left*

11 *zzz) Location*

12 *aaaa) MultiColumn*

13 *bbbb) ToString*

14 *Description*

15 Gets or sets a value indicating whether the **System.Windows.Forms.ListBox**  
16 supports multiple columns.

17 A multicolumn list box places items into as many columns as are needed to make  
18 vertical scrolling unnecessary. The user can use the keyboard to navigate to  
19 columns that are not currently visible. Set the  
20 **System.Windows.Forms.ListBox.HorizontalScrollbar** property to **true** to  
21 display a horizontal scroll bar that allows the user to scroll to columns that are  
22 not currently shown in the visible region of the  
23 **System.Windows.Forms.ListBox** . The value of the  
24 **System.Windows.Forms.ListBox.ColumnWidth** property determines the  
25 width of each column.

cccc) *Name*

dddd) *Parent*

eeee) *PreferredHeight*

ffff) *ToString*

*Description*

Gets the combined height of all items in the **System.Windows.Forms.ListBox**

This property enables you to determine the height that the **System.Windows.Forms.ListBox** needs to be sized to in order to display every available item in the list and to avoid displaying vertical scroll bars. If the amount of items in the **System.Windows.Forms.ListBox** is large, sizing the control using the value of the **System.Windows.Forms.ListBox.PreferredHeight** property might cause the **System.Windows.Forms.ListBox** to be sized outside of the client area of the form or container it is located within.

*gggg) ProductName*

*hhhh) ProductVersion*

*iiii) RecreatingHandle*

*jjjj) Region*

*kkkk) RenderRightToLeft*

*llll) ResizeRedraw*

*mmmm)Right*

*nnnn) RightToLeft*

*oooo) ToString*

### *Description*

Gets or sets a value indicating whether text displayed by the control is displayed from right to left.

This property allows your menus to support languages that are written from right to left. When this property is set to **true** , item text is displayed from right to left instead of the default left to right method.

*pppp) ScrollAlwaysVisible*

*qqqq) ToString*

[C#]        public        bool        ScrollAlwaysVisible        {get;        set;}

[C++] public: \_\_property bool get\_ScrollAlwaysVisible();public: \_\_property void  
set\_ScrollAlwaysVisible(bool);

[VB]        Public        Property        ScrollAlwaysVisible        As        Boolean

[JScript] public function get ScrollAlwaysVisible() : Boolean;public function set

1 ScrollAlwaysVisible(Boolean);

3 *Description*

4 Gets or sets a value indicating whether the vertical scroll bar is shown at all  
5 times.

6 The **System.Windows.Forms.ListBox.ScrollAlwaysVisible** property  
7 indicates whether a vertical scroll bar is always displayed, even if the number of  
8 items in the **System.Windows.Forms.ListBox** does not require displaying the  
9 vertical scroll bar. By default, a **System.Windows.Forms.ListBox** only shows  
10 a vertical scroll bar when there are enough items to warrant displaying. For  
11 multicolumn list boxes, the

12 **System.Windows.Forms.ListBox.ScrollAlwaysVisible** property indicates  
13 that a horizontal scroll bar is displayed. A vertical scroll bar is never displayed  
14 regardless of the value of this property for a multicolumn  
15 **System.Windows.Forms.ListBox**.

16 *rrrr) SelectedIndex*

17 *ssss) ToString*

18 [C#] public override int SelectedIndex {get; set;}

19 [C++] public: \_\_property virtual int get\_SelectedIndex();public: \_\_property virtual  
20 void set\_SelectedIndex(int);

21 [VB] Overrides Public Property SelectedIndex As Integer

22 [JScript] public function get SelectedIndex() : int;public function set  
23 SelectedIndex(int);

24 *Description*

25 Gets or sets the zero-based index of the currently selected item in a  
26 **System.Windows.Forms.ListBox**.

27 For a standard **System.Windows.Forms.ListBox**, you can use this property  
28 to determine the index of the item that is selected in the  
29 **System.Windows.Forms.ListBox**. If the

**System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is set to either **SelectionMode.MultiSimple** or **SelectionMode.MultiExtended** (which indicates a multiple-selection **System.Windows.Forms.ListBox** ) and multiple items are selected in the list, this property can return the index to any selected item.

*tttt) SelectedIndices*

*uuuu) ToString*

```
[C#]    public    ListBox.SelectedIndexCollection    SelectedIndices    {get;}
```

```
[C++]    public:    __property    ListBox.SelectedIndexCollection*  
get_SelectedIndices();
```

```
[VB]    Public    ReadOnly    Property    SelectedIndices    As  
ListBox.SelectedIndexCollection
```

```
[JScript] public function get SelectedIndices() : ListBox.SelectedIndexCollection;
```

#### *Description*

Gets a collection that contains the zero-based indices of all currently selected items in the **System.Windows.Forms.ListBox** .

For a multiple-selection **System.Windows.Forms.ListBox** , this property returns a collection containing the indices to all items that are selected in the **System.Windows.Forms.ListBox** . For a single-selection **System.Windows.Forms.ListBox** , this property returns a collection containing a single element containing the index of the only selected item in the **System.Windows.Forms.ListBox** . For more information on how to manipulate the members of the selected indices collection, see **System.Windows.Forms.ListBox.SelectedIndexCollection** .

vvvv) *SelectedItem*

www) *ToString*

[C#]        public        object        SelectedItem        {get;        set;}

[C++] public: \_\_property Object\* get\_SelectedItem();public: \_\_property void  
set\_SelectedItem(Object\*);

[VB]        Public        Property        SelectedItem        As        Object

[JScript] public function get SelectedItem() : Object;public function set  
SelectedItem(Object);

### *Description*

Gets or sets the currently selected item in the  
**System.Windows.Forms.ListBox** .

For a standard **System.Windows.Forms.ListBox** , you can use this property to determine which item is selected in the **System.Windows.Forms.ListBox** . If the **System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is set to either **SelectionMode.MultiSimple** or **SelectionMode.MultiExtended** (which indicates a multiple-selection **System.Windows.Forms.ListBox** ) and multiple items are selected in the list, this property can return any selected item.

xxxx) *SelectedItems*

yyyy) *ToString*

[C#]        public        ListBox.SelectedObjectCollection        SelectedItems        {get;}

[C++] public: \_\_property ListBox.SelectedObjectCollection\* get\_SelectedItems();

[VB]        Public        ReadOnly        Property        SelectedItems        As  
ListBox.SelectedObjectCollection

[JScript] public function get SelectedItems() : ListBox.SelectedObjectCollection;

1  
2 *Description*

3 Gets a collection containing the currently selected items in the  
4 **System.Windows.Forms.ListBox** .

5 For a multiple-selection **System.Windows.Forms.ListBox** , this property  
6 returns a collection containing the indices to all items that are selected in the  
7 **System.Windows.Forms.ListBox** . For a single-selection  
8 **System.Windows.Forms.ListBox** , this property returns a collection  
9 containing a single element containing the index of the only selected item in the  
10 **System.Windows.Forms.ListBox** . For more information on how to  
11 manipulate the members of the selected indices collection, see  
12 **System.Windows.Forms.ListBox.SelectedIndexCollection** .

13  
14 *zzzz) SelectedValue*

15  
16 *aaaaa) SelectionMode*

17  
18 *bbbbbb) ToString*

19  
20 *Description*

21 Gets or sets the method in which items are selected in the  
22 **System.Windows.Forms.ListBox** .

23 The **System.Windows.Forms.ListBox.SelectionMode** property enables you  
24 to determine how many items in the **System.Windows.Forms.ListBox** a user  
25 can select at one time and how the user can make multiple-selections. When the  
**System.Windows.Forms.ListBox.SelectionMode** property is set to  
**SelectionMode.MultiExtended** , pressing SHIFT and clicking the mouse or  
pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT  
ARROW, and RIGHT ARROW) extends the selection from the previously selected  
item to the current item. Pressing CTRL and clicking the mouse selects or  
deselects an item in the list. When the property is set to  
**SelectionMode.MultiSimple** , a mouse click or pressing the SPACEBAR selects  
or deselects an item in the list.



1        *cccc) ShowFocusCues*

2        *dddd) ShowKeyboardCues*

3        *eeee) Site*

4        *ffff) Size*

5        *gggg) Sorted*

6        *hhhh) ToString*

7  
8  
9        *Description*

10       Gets or sets a value indicating whether the items in the  
11       **System.Windows.Forms.ListBox** are sorted alphabetically.

12       You can use this property to automatically sort items alphabetically in a  
13       **System.Windows.Forms.ListBox** . As items are added to a sorted  
14       **System.Windows.Forms.ListBox** , the items are moved to the appropriate  
15       location in the sorted list. When adding items to a  
16       **System.Windows.Forms.ListBox** , it is more efficient to sort the items first  
17       and then add new items.

18        *iiii) TabIndex*

19        *jjjj) TabStop*

20        *kkkk) Tag*

21        *llll) Text*

22        *mmmm) ToString*

23       *Description*

24       Gets or searches for the text of the currently selected item in the  
25       **System.Windows.Forms.ListBox** .

When the value of this property is set to a string value, the **System.Windows.Forms.ListBox** searches for the item within the **System.Windows.Forms.ListBox** that matches the specified text and selects the item. You can also use this property to determine which items are currently selected in the **System.Windows.Forms.ListBox** . If the **System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is set to **SelectionMode.MultiExtended** , this property returns the text of the first selected item. If the **System.Windows.Forms.ListBox.SelectionMode** property of the **System.Windows.Forms.ListBox** is not set to **SelectionMode.None** , this property returns the text of the first selected item.

*nnnnn)Top*

*ooooo) TopIndex*

*ppppp) ToString*

### *Description*

Gets or sets the index of the first visible item in the **System.Windows.Forms.ListBox** .

Initially, the item with the index position zero (0) is at the top of the visible region of the **System.Windows.Forms.ListBox** . If the contents of the **System.Windows.Forms.ListBox** have been scrolled, another item might be at the top of the control's display area. You can use this property to obtain the index within the **System.Windows.Forms.ListBox.ObjectCollection** for the **System.Windows.Forms.ListBox** of the item that is currently positioned at the top of the visible region of the control. You can also use this property to position an item in the list at the top of the visible region of the control.

1        *qqqqq) TopLevelControl*

2        *rrrrr) UseTabStops*

3        *sssss) ToString*

4  
5  
6        *Description*

7        Gets or sets a value indicating whether the **System.Windows.Forms.ListBox**  
8        can recognize and expand tab characters when drawing its strings.

9        *ttttt) ValueMember*

10       *uuuuu) Visible*

11       *vvvvv) Width*

12       *wwwww) WindowTarget*

13       *xxxxx) ToString*

14       *yyyyy) ToString*

15  
16  
17       *Description*

18       Occurs when a visual aspect of an owner-drawn  
19       **System.Windows.Forms.ListBox** changes.

20       This event is used by an owner-drawn **System.Windows.Forms.ListBox** . The  
21       event is only raised when the **System.Windows.Forms.ListBox.DrawMode**  
22       property is set to **DrawMode.OwnerDrawFixed** or  
23       **DrawMode.OwnerDrawVariable** . You can use this event to perform the  
24       tasks needed to draw items in the **System.Windows.Forms.ListBox** . If you  
25       have a variable-sized item (when the  
26       **System.Windows.Forms.ListBox.DrawMode** property is set to  
27       **DrawMode.OwnerDrawVariable** ), before drawing an item, the  
28       **System.Windows.Forms.ListBox.MeasureItem** event is raised. You can  
29       create an event handler for the  
30       **System.Windows.Forms.ListBox.MeasureItem** event to specify the size for

event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .

ItemActivation enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the user action that is required to activate items in a list view control and the feedback that is given as the user moves the mouse pointer over an item.

Use the members of this enumeration to set the value of the **System.Windows.Forms.ListView.Activation** property of the **System.Windows.Forms.ListView** control.

*b) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | ItemActivation | OneClick;         |
| [C++]     | public: | const | ItemActivation | OneClick;         |
| [VB]      | Public  | Const | OneClick       | As ItemActivation |
| [JScript] | public  | var   | OneClick       | : ItemActivation; |

*Description*

The user must single-click to activate items. The cursor changes to a hand pointer cursor, and the item text changes color as the user moves the mouse pointer over the item.

*c) ToString*

|       |         |       |                |           |
|-------|---------|-------|----------------|-----------|
| [C#]  | public  | const | ItemActivation | Standard; |
| [C++] | public: | const | ItemActivation | Standard; |

|           |        |       |          |    |                 |
|-----------|--------|-------|----------|----|-----------------|
| [VB]      | Public | Const | Standard | As | ItemActivation  |
| [JScript] | public | var   | Standard | :  | ItemActivation; |

*Description*

The user must double-click to activate items. No feedback is given as the user moves the mouse pointer over an item.

*d) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | ItemActivation | TwoClick;         |
| [C++]     | public: | const | ItemActivation | TwoClick;         |
| [VB]      | Public  | Const | TwoClick       | As ItemActivation |
| [JScript] | public  | var   | TwoClick       | : ItemActivation; |

*Description*

The user must double-click to activate items and the item text changes color as the user moves the mouse pointer over the item.

ItemBoundsPortion enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies a portion of the list view item from which to retrieve the bounding rectangle.

Use the members of this enumeration when calling the **System.Windows.Forms.ListView.GetItemRect(System.Int32)** method of the **System.Windows.Forms.ListView** control. This enumeration is also used when calling the **System.Windows.Forms.ListViewItem.GetBounds(System.Windows.For**

**ms.ItemBoundsPortion)** method of the **System.Windows.Forms.ListViewItem** class.

**b) ToString**

|           |         |       |                   |                      |
|-----------|---------|-------|-------------------|----------------------|
| [C#]      | public  | const | ItemBoundsPortion | Entire;              |
| [C++]     | public: | const | ItemBoundsPortion | Entire;              |
| [VB]      | Public  | Const | Entire            | As ItemBoundsPortion |
| [JScript] | public  | var   | Entire            | : ItemBoundsPortion; |

*Description*

The bounding rectangle of the entire item, including the icon, the item text, and the subitem text (if displayed), should be retrieved.

**c) ToString**

|           |         |       |                   |                      |
|-----------|---------|-------|-------------------|----------------------|
| [C#]      | public  | const | ItemBoundsPortion | Icon;                |
| [C++]     | public: | const | ItemBoundsPortion | Icon;                |
| [VB]      | Public  | Const | Icon              | As ItemBoundsPortion |
| [JScript] | public  | var   | Icon              | : ItemBoundsPortion; |

*Description*

The bounding rectangle of the icon or small icon should be retrieved.

**d) ToString**

|       |         |       |                   |                      |
|-------|---------|-------|-------------------|----------------------|
| [C#]  | public  | const | ItemBoundsPortion | ItemOnly;            |
| [C++] | public: | const | ItemBoundsPortion | ItemOnly;            |
| [VB]  | Public  | Const | ItemOnly          | As ItemBoundsPortion |

1 [JScript] public var ItemOnly : ItemBoundsPortion;

3 *Description*

4 The bounding rectangle of the icon or small icon and the item text should be  
5 retrieved. In all views except Report view of the  
6 **System.Windows.Forms.ListView** , this value specifies the same bounding  
rectangle as the **Entire** value. In Report view, this value specifies the bounding  
rectangle specified by the **Entire** value without the subitems.

7 e) *ToString*

9 [C#] public const ItemBoundsPortion Label;

10 [C++] public: const ItemBoundsPortion Label;

11 [VB] Public Const Label As ItemBoundsPortion

12 [JScript] public var Label : ItemBoundsPortion;

14 *Description*

15 The bounding rectangle of the item text should be retrieved.

16 ItemChangedEventArgs class (System.Windows.Forms)

17 a) *ToString*

20 *Description*

21 Provides data for the  
22 **System.Windows.Forms.CurrencyManager.ItemChanged** event.

23 An **System.Windows.Forms.CurrencyManager.ItemChanged** event occurs  
24 whenever the item in a list is changed. For example, this event will occur when  
the text of the list item is changed to a new value. This event is not raised when  
25 the item is moved to a new position within the list because of a new item being  
added.

b) *Index*

c) *ToString*

|           |         |            |          |                  |
|-----------|---------|------------|----------|------------------|
| [C#]      | public  | int        | Index    | {get;}           |
| [C++]     | public: | __property | int      | get_Index();     |
| [VB]      | Public  | ReadOnly   | Property | Index As Integer |
| [JScript] | public  | function   | get      | Index() : int;   |

#### *Description*

Indicates the position of the item being changed within the list.

ItemChangedEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **System.Windows.Forms.CurrencyManager.ItemChanged** event of the **System.Windows.Forms.CurrencyManager** class. The source of the event. An **System.Windows.Forms.ItemChangedEventArgs** that contains the event data.

When you create an **System.Windows.Forms.ItemChangedEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .



ItemCheckEventArgs class (System.Windows.Forms)

a) *ToString*

### *Description*

Provides data for the **System.Windows.Forms.CheckedListBox.ItemCheck** event of the **System.Windows.Forms.CheckedListBox** and **System.Windows.Forms.ListView** controls.

The **System.Windows.Forms.CheckedListBox.ItemCheck** event occurs when the checked state of an item in a checked list box changes. The **System.Windows.Forms.ItemCheckEventArgs** class specifies the index of the item to change, the current value of the check box for the item, and the new value to set for the check box.

b) *ItemCheckEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#] public ItemCheckEventArgs(int index, CheckState newCheckValue,
                                CheckState                               currentValue);
```

```
[C++] public: ItemCheckEventArgs(int index, CheckState newCheckValue,
                                CheckState                               currentValue);
```

```
[VB] Public Sub New(ByVal index As Integer, ByVal newCheckValue As
CheckState,          ByVal          currentValue          As          CheckState)
```

```
[JScript] public function ItemCheckEventArgs(index : int, newCheckValue :
CheckState,          currentValue          :          CheckState);
```

### *Description*

Initializes a new instance of the **System.Windows.Forms.ItemCheckEventArgs** class. The zero-based index of the item to change. One of the **System.Windows.Forms.CheckState** values that indicates whether to change the check box for the item to be checked, unchecked, or indeterminate. One of the **System.Windows.Forms.CheckState** values that indicates whether the check box for the item is currently checked, unchecked, or indeterminate.

d) *CurrentValue*

e) *ToString*

```
[C#]      public      CheckState      CurrentValue      {get;}
[C++]      public:      __property      CheckState      get_CurrentValue();
[VB]      Public      ReadOnly      Property      CurrentValue      As      CheckState
[JScript]      public      function      get      CurrentValue()      :      CheckState;
```

### *Description*

Gets a value indicating the current state of the item's check box.

This property enables you to determine the check state of the specified item in the **System.Windows.Forms.CheckedListBox** before the check state change to apply is made.

f) *Index*

g) *ToString*

```
[C#]      public      int      Index      {get;}
[C++]      public:      __property      int      get_Index();
[VB]      Public      ReadOnly      Property      Index      As      Integer
[JScript]      public      function      get      Index()      :      int;
```

## Description

Gets the zero-based index of the item to change.

You can use this property to determine which item's check box in the **System.Windows.Forms.CheckedListBox** is being changed.

*h) NewValue*

*i) ToString*

```
[C#]      public      CheckState      NewValue      {get;      set;}
```

```
[C++] public: __property CheckState get_NewValue();public: __property void  
set_NewValue(CheckState);
```

```
[VB]      Public      Property      NewValue      As      CheckState
```

```
[JScript] public function get NewValue() : CheckState;public function set  
NewValue(CheckState);
```

## Description

Gets or sets a value indicating whether to set the check box for the item to be checked, unchecked, or indeterminate.

This property enables you to determine the new check state for the specified item before the check state is changed by the **System.Windows.Forms.CheckedListBox** control. In addition to determining the new check state, you can use this property in an event handler for the **System.Windows.Forms.CheckedListBox.ItemCheck** event to change the state to a different check state than the one specified. For example, if the user placed a check mark next to an item in the **System.Windows.Forms.CheckedListBox** that you have determined should not be checked based on the state of your application, you can override the change in the check mark state by setting this property to its previous setting or to a different check state.

ItemCheckEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **ItemCheck** event of a **System.Windows.Forms.CheckedListBox** or **System.Windows.Forms.ListView** control. The source of the event. An **System.Windows.Forms.ItemCheckEventArgs** that contains the event data.

When you create an **System.Windows.Forms.ItemCheckEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ItemDragEventArgs class (System.Windows.Forms)

a) *ToString*

#### *Description*

Provides data for the **System.Windows.Forms.ListView.ItemDrag** event of the **System.Windows.Forms.ListView** and **System.Windows.Forms.TreeView** controls.

The **System.Windows.Forms.ListView.ItemDrag** event occurs when the user begins dragging an item. An **System.Windows.Forms.ItemDragEventArgs** object specifies which mouse button was pressed.

b) *ItemDragEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#]      public      ItemDragEventArgs(MouseButtons      button);
[C++]      public:      ItemDragEventArgs(MouseButtons      button);
[VB]      Public      Sub      New(ByVal      button      As      MouseButtons)
[JavaScript] public function ItemDragEventArgs(button : MouseButtons); Initializes a
new instance of the System.Windows.Forms.ItemDragEventArgs class.
```

*Description*

Initializes a new instance of the **System.Windows.Forms.ItemDragEventArgs** class with a specified mouse button.

When this version of the constructor is used, the value of the **System.Windows.Forms.ItemDragEventArgs.Item** property is automatically set to **null** . One of the **System.Windows.Forms.MouseButtons** values that represents which mouse button was pressed.

d) *ItemDragEventArgs*

*Example Syntax:*

e) *ToString*

```
[C#]      public      ItemDragEventArgs(MouseButtons      button,      object      item);
[C++]      public:      ItemDragEventArgs(MouseButtons      button,      Object*      item);
[VB]      Public      Sub      New(ByVal      button      As      MouseButtons, ByVal      item      As      Object)
[JavaScript] public function ItemDragEventArgs(button : MouseButtons, item :
Object);
```

*Description*

Initializes a new instance of the

**System.Windows.Forms.ItemDragEventArgs** class with a specified mouse button and the item that is being dragged. One of the **System.Windows.Forms.MouseButtons** values that represents which mouse button was pressed. The item being dragged.

f) *Button*

g) *ToString*

[C#]            public            MouseButtons            Button            {get;}

[C++]           public:            \_\_property            MouseButtons            get\_Button();

[VB]           Public           ReadOnly           Property           Button           As           MouseButtons

[JScript]       public           function           get           Button()           :           MouseButtons;

#### *Description*

Gets the name of the mouse button that was clicked during the drag operation.

This property enables you to determine which mouse button was pressed during a drag and drop operation. The value of this property can be used to properly determine how the drag and drop operation should be performed.

h) *Item*

i) *ToString*

[C#]            public            object            Item            {get;}

[C++]           public:            \_\_property            Object\*            get\_Item();

[VB]           Public           ReadOnly           Property           Item           As           Object

[JScript]       public           function           get           Item()           :           Object;

#### *Description*

Gets the item that is being dragged.

1 You can use this property to determine which item from the  
2 **System.Windows.Forms.TreeView** or **System.Windows.Forms.ListView**  
controls is being dragged from the control.

3 ItemDragEventHandler delegate (System.Windows.Forms)

4 a) *ToString*

5  
6  
7 *Description*

8 Represents the method that will handle the  
9 **System.Windows.Forms.ListView.ItemDrag** event of a  
10 **System.Windows.Forms.ListView** or **System.Windows.Forms.TreeView**  
control. The source of the event. An  
11 **System.Windows.Forms.ItemDragEventArgs** that contains the event data.

12 When you create an **System.Windows.Forms.ItemDragEventHandler**  
13 delegate, you identify the method that will handle the event. To associate the  
14 event with your event handler, add an instance of the delegate to the event. The  
15 event handler is called whenever the event occurs, unless you remove the  
16 delegate. For more information about event handler delegates, see .

17  
18 IWin32Window interface (System.Windows.Forms)

19 a) *ToString*

20  
21  
22 *Description*

23 Provides an interface to expose Win32 HWND handles.

24 This interface is implemented on objects that expose Win32 HWND handles. The  
25 resultant handle can be used with Win32 API calls.

26 b) *Handle*

27 c) *ToString*

28 [C#]                      IntPtr                      Handle                      {get;}

|           |          |          |        |          |               |
|-----------|----------|----------|--------|----------|---------------|
| [C++]     |          |          | IntPtr |          | get_Handle(); |
| [VB]      | ReadOnly | Property | Handle | As       | IntPtr        |
| [JScript] | abstract | function | get    | Handle() | : IntPtr;     |

#### Description

Gets the handle to the window represented by the implementer.

Depending on the implementer, the value of the **System.Windows.Forms.IWin32Window.Handle** property could change during the life of the window.

IWindowTarget interface (System.Windows.Forms)

#### a) ToString

#### Description

This interface defines the communication layer between a Control object and the Win32 API. Each Control object has an internal implementation this interface that is called by the Win32 window.

#### b) OnHandleChange

|           |          |                                |             |
|-----------|----------|--------------------------------|-------------|
| [C#]      | void     | OnHandleChange(IntPtr          | newHandle); |
| [C++]     | void     | OnHandleChange(IntPtr          | newHandle); |
| [VB]      | Sub      | OnHandleChange(ByVal newHandle | As IntPtr)  |
| [JScript] | function | OnHandleChange(newHandle       | : IntPtr);  |

#### Description

Called when the window handle of the control has changed. The new window handle value.



c) *OnMessage*

```
[C#]          void          OnMessage(ref          Message          m);
[C++]          void          OnMessage(Message*          m);
[VB]          Sub          OnMessage(ByRef          m          As          Message)
[JScript]          function          OnMessage(m          :          Message);
```

*Description*

Called to do control-specific processing for this window. The message to process.

KeyEventArgs class (System.Windows.Forms)

a) *OnMessage*

*Description*

Provides data for the **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

A **System.Windows.Forms.KeyEventArgs** , which specifies the key the user pressed and whether any modifier keys (CTRL, ALT, and SHIFT) were pressed at the same time, is passed with each **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

b) *KeyEventArgs*

*Example Syntax:*

c) *OnMessage*

```
[C#]          public          KeyEventArgs(Keys          keyData);
[C++]          public:          KeyEventArgs(Keys          keyData);
```

1 [VB] Public Sub New(ByVal keyData As Keys)

2 [JScript] public function KeyEventArgs(keyData : Keys);

4 *Description*

5 Initializes a new instance of the **System.Windows.Forms.KeyEventArgs**  
6 class. The key code constant for the key that was pressed, combined with any  
7 modifier flags that indicate which CTRL, SHIFT, and ALT keys were pressed at  
the same time. Possible values are obtained by applying the bitwise OR (|)  
operator to constants from **System.Windows.Forms.Keys**.

8 d) *Alt*

9 e) *OnMessage*

11 [C#] public virtual bool Alt {get;}

12 [C++] public: \_\_property virtual bool get\_Alt();

13 [VB] Overridable Public ReadOnly Property Alt As Boolean

14 [JScript] public function get Alt() : Boolean;

16 *Description*

17 Gets a value indicating whether the ALT key was pressed.

18 f) *Control*

19 g) *OnMessage*

21 [C#] public bool Control {get;}

22 [C++] public: \_\_property bool get\_Control();

23 [VB] Public ReadOnly Property Control As Boolean

24 [JScript] public function get Control() : Boolean;

*Description*

Gets a value indicating whether the CTRL key was pressed.

*h) Handled*

*i) OnMessage*

[C#]            public            bool            Handled            {get;            set;}

[C++]   public:   \_\_property   bool   get\_Handled();public:   \_\_property   void  
set\_Handled(bool);

[VB]           Public           Property           Handled           As           Boolean

[JScript]   public   function   get   Handled() : Boolean;public   function   set  
Handled(Boolean);

*Description*

Gets or sets a value indicating whether the event was handled.

*j) KeyCode*

*k) OnMessage*

[C#]            public            Keys            KeyCode            {get;}

[C++]           public:            \_\_property            Keys            get\_KeyCode();

[VB]           Public           ReadOnly           Property           KeyCode           As           Keys

[JScript]   public           function           get           KeyCode()           :           Keys;

*Description*

Gets the keyboard code for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

l) *KeyData*

m) *OnMessage*

|           |         |            |          |                   |
|-----------|---------|------------|----------|-------------------|
| [C#]      | public  | Keys       | KeyData  | {get;}            |
| [C++]     | public: | __property | Keys     | get_KeyData();    |
| [VB]      | Public  | ReadOnly   | Property | KeyData As Keys   |
| [JScript] | public  | function   | get      | KeyData() : Keys; |

#### *Description*

Gets the key data for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

You can use constants from **System.Windows.Forms.Keys** to extract information from the **System.Windows.Forms.KeyEventArgs.KeyData** property. Use the bitwise AND (&) operator to compare data returned by **System.Windows.Forms.KeyEventArgs.KeyData** with constants in **System.Windows.Forms.Keys** to obtain information about which keys the user pressed. To determine whether a specific modifier key was pressed, use the **System.Windows.Forms.KeyEventArgs.Control**, **System.Windows.Forms.KeyEventArgs.Shift**, and **System.Windows.Forms.KeyEventArgs.Alt** properties.

n) *KeyValue*

o) *OnMessage*

|           |         |            |          |                     |
|-----------|---------|------------|----------|---------------------|
| [C#]      | public  | int        | KeyValue | {get;}              |
| [C++]     | public: | __property | int      | get_KeyValue();     |
| [VB]      | Public  | ReadOnly   | Property | KeyValue As Integer |
| [JScript] | public  | function   | get      | KeyValue() : int;   |

*Description*

Gets the keyboard value for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event.

*p) Modifiers*

*q) OnMessage*

|           |         |            |           |                     |
|-----------|---------|------------|-----------|---------------------|
| [C#]      | public  | Keys       | Modifiers | {get;}              |
| [C++]     | public: | __property | Keys      | get_Modifiers();    |
| [VB]      | Public  | ReadOnly   | Property  | Modifiers As Keys   |
| [JScript] | public  | function   | get       | Modifiers() : Keys; |

*Description*

Gets the modifier flags for a **System.Windows.Forms.Control.KeyDown** or **System.Windows.Forms.Control.KeyUp** event. This indicates which modifier keys (CTRL, SHIFT, and/or ALT) were pressed.

To determine whether a specific modifier key was pressed, use the **System.Windows.Forms.KeyEventArgs.Control**, **System.Windows.Forms.KeyEventArgs.Shift**, and **System.Windows.Forms.KeyEventArgs.Alt** properties. Modifier flags can be combined with bitwise OR.

*r) Shift*

*s) OnMessage*

|           |             |            |          |          |                  |
|-----------|-------------|------------|----------|----------|------------------|
| [C#]      | public      | virtual    | bool     | Shift    | {get;}           |
| [C++]     | public:     | __property | virtual  | bool     | get_Shift();     |
| [VB]      | Overridable | Public     | ReadOnly | Property | Shift As Boolean |
| [JScript] | public      | function   | get      | Shift()  | : Boolean;       |

1  
2 *Description*

3 Gets a value indicating whether the SHIFT key was pressed.

4 KeyEventHandler delegate (System.Windows.Forms)

5 a) *ToString*

6  
7  
8 *Description*

9 Represents a method that will handle the  
10 **System.Windows.Forms.Control.KeyUp** or  
11 **System.Windows.Forms.Control.KeyDown** event of a  
12 **System.Windows.Forms.Control** . The source of the event. A  
13 **System.Windows.Forms.KeyEventArgs** that contains the event data.

14 When you create a **System.Windows.Forms.KeyEventHandler** delegate, you  
15 identify the method that will handle the event. To associate the event with your  
16 event handler, add an instance of the delegate to the event. The event handler is  
17 called whenever the event occurs, unless you remove the delegate. For more  
18 information about handling events with delegates, see .

19 KeyPressEventArgs class (System.Windows.Forms)

20 a) *ToString*

21  
22  
23  
24  
25 *Description*

Provides data for the **System.Windows.Forms.Control.KeyPress** event.

A **System.Windows.Forms.KeyPressEventArgs** specifies the character that  
is composed when the user presses a key. For example, when the user presses  
SHIFT + K, the **System.Windows.Forms.KeyPressEventArgs.KeyChar**  
property returns an uppercase K.

b) *KeyPressEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#]          public          KeyPressEventArgs(char          keyChar);
[C++]         public:         KeyPressEventArgs(__wchar_t      keyChar);
[VB]          Public          Sub          New(ByVal          keyChar          As          Char)
[JScript]     public          function    KeyPressEventArgs(keyChar      :      Char);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.KeyPressEventArgs** class. The ASCII character corresponding to the key the user pressed.

d) *Handled*

e) *ToString*

```
[C#]          public          bool          Handled          {get;          set;}
[C++]         public:  __property  bool  get_Handled();public:  __property  void
set_Handled(bool);
[VB]          Public          Property          Handled          As          Boolean
[JScript]     public          function  get  Handled() : Boolean;public          function  set
Handled(Boolean);
```

*Description*

Gets or sets a value indicating whether the **System.Windows.Forms.Control.KeyPress** event was handled.

If the event is not handled, it will be sent to Windows for default processing.

f) *KeyChar*

g) *ToString*

|           |         |            |               |                 |
|-----------|---------|------------|---------------|-----------------|
| [C#]      | public  | char       | KeyChar       | {get;}          |
| [C++]     | public: | __property | __wchar_t     | get_KeyChar();  |
| [VB]      | Public  | ReadOnly   | Property      | KeyChar As Char |
| [JScript] | public  | function   | get KeyChar() | : Char;         |

*Description*

Gets the character corresponding to the key pressed.

KeyPressEventHandler delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents a method that will handle the **System.Windows.Forms.Control.KeyPress** event of a **System.Windows.Forms.Control** . The source of the event. A **System.Windows.Forms.KeyPressEventArgs** that contains the event data.

When you create a **System.Windows.Forms.KeyPressEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about handling events with delegates, see .



Keys enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies key codes and modifiers.

This class contains constants to use for processing keyboard input. Keys are identified by key values, which consist of a key code and a set of modifiers combined into a single integer value. The four left digits of a key value contain the key code (which is the same as a Windows virtual key code). The four right digits of a key value contain modifier bits for the SHIFT, CONTROL, and ALT keys.

*b) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | A;      |
| [C++]     | public: | const | Keys | A;      |
| [VB]      | Public  | Const | A    | As Keys |
| [JScript] | public  | var   | A    | : Keys; |

*Description*

The A key.

*c) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Add;    |
| [C++]     | public: | const | Keys | Add;    |
| [VB]      | Public  | Const | Add  | As Keys |
| [JScript] | public  | var   | Add  | : Keys; |

*Description*

The Add key.

*d) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Alt;    |
| [C++]     | public: | const | Keys | Alt;    |
| [VB]      | Public  | Const | Alt  | As Keys |
| [JScript] | public  | var   | Alt  | : Keys; |

*Description*

The ALT modifier key.

*e) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Apps;   |
| [C++]     | public: | const | Keys | Apps;   |
| [VB]      | Public  | Const | Apps | As Keys |
| [JScript] | public  | var   | Apps | : Keys; |

*Description*

The Application key (Microsoft Natural Keyboard).

*f) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Attn; |
| [C++] | public: | const | Keys | Attn; |

|   |           |        |       |      |    |       |
|---|-----------|--------|-------|------|----|-------|
| 1 | [VB]      | Public | Const | Attn | As | Keys  |
| 2 | [JScript] | public | var   | Attn | :  | Keys; |

*Description*

The ATTN key.

*g) ToString*

|    |           |         |       |   |      |       |
|----|-----------|---------|-------|---|------|-------|
| 8  | [C#]      | public  | const |   | Keys | B;    |
| 9  | [C++]     | public: | const |   | Keys | B;    |
| 10 | [VB]      | Public  | Const | B | As   | Keys  |
| 11 | [JScript] | public  | var   | B | :    | Keys; |

*Description*

The B key.

*h) ToString*

|    |           |         |       |      |      |       |
|----|-----------|---------|-------|------|------|-------|
| 17 | [C#]      | public  | const |      | Keys | Back; |
| 18 | [C++]     | public: | const |      | Keys | Back; |
| 19 | [VB]      | Public  | Const | Back | As   | Keys  |
| 20 | [JScript] | public  | var   | Back | :    | Keys; |

*Description*

The BACKSPACE key.

*i) ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | BrowserBack; |
| [C++]     | public: | const | Keys        | BrowserBack; |
| [VB]      | Public  | Const | BrowserBack | As Keys      |
| [JScript] | public  | var   | BrowserBack | : Keys;      |

*Description*

The Browser Back key (Windows 2000 or later).

*j) ToString*

|           |         |       |                  |                   |
|-----------|---------|-------|------------------|-------------------|
| [C#]      | public  | const | Keys             | BrowserFavorites; |
| [C++]     | public: | const | Keys             | BrowserFavorites; |
| [VB]      | Public  | Const | BrowserFavorites | As Keys           |
| [JScript] | public  | var   | BrowserFavorites | : Keys;           |

*Description*

The Browser Favorites key (Windows 2000 or later).

*Description*

The Browser Favorites key.

*k) ToString*

|       |         |       |                |                 |
|-------|---------|-------|----------------|-----------------|
| [C#]  | public  | const | Keys           | BrowserForward; |
| [C++] | public: | const | Keys           | BrowserForward; |
| [VB]  | Public  | Const | BrowserForward | As Keys         |

[JScript] public var BrowserForward : Keys;

*Description*

The Browser Forward key (Windows 2000 or later).

*l) ToString*

[C#] public const Keys BrowserHome;

[C++] public: const Keys BrowserHome;

[VB] Public Const BrowserHome As Keys

[JScript] public var BrowserHome : Keys;

*Description*

The Browser Home key (Windows 2000 or later).

*m) ToString*

[C#] public const Keys BrowserRefresh;

[C++] public: const Keys BrowserRefresh;

[VB] Public Const BrowserRefresh As Keys

[JScript] public var BrowserRefresh : Keys;

*Description*

The Browser Refresh key (Windows 2000 or later).

*n) ToString*

[C#] public const Keys BrowserSearch;

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C++]     | public: | const | Keys          | BrowserSearch; |
| [VB]      | Public  | Const | BrowserSearch | As Keys        |
| [JScript] | public  | var   | BrowserSearch | : Keys;        |

*Description*

The Browser Search key (Windows 2000 or later).

*o) ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | BrowserStop; |
| [C++]     | public: | const | Keys        | BrowserStop; |
| [VB]      | Public  | Const | BrowserStop | As Keys      |
| [JScript] | public  | var   | BrowserStop | : Keys;      |

*Description*

The Browser Stop key (Windows 2000 or later).

*p) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | C;      |
| [C++]     | public: | const | Keys | C;      |
| [VB]      | Public  | Const | C    | As Keys |
| [JScript] | public  | var   | C    | : Keys; |

*Description*

The C key.

**q) ToString**

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Cancel; |
| [C++]     | public: | const | Keys   | Cancel; |
| [VB]      | Public  | Const | Cancel | As Keys |
| [JScript] | public  | var   | Cancel | : Keys; |

*Description*

The CANCEL key.

**r) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Capital; |
| [C++]     | public: | const | Keys    | Capital; |
| [VB]      | Public  | Const | Capital | As Keys  |
| [JScript] | public  | var   | Capital | : Keys;  |

*Description*

The CAPS LOCK key.

**s) ToString**

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | CapsLock; |
| [C++]     | public: | const | Keys     | CapsLock; |
| [VB]      | Public  | Const | CapsLock | As Keys   |
| [JScript] | public  | var   | CapsLock | : Keys;   |

1  
2 *Description*

3 The CAPS LOCK key.

4 *t) ToString*

5  
6 [C#] public const Keys Clear;  
7 [C++] public: const Keys Clear;  
8 [VB] Public Const Clear As Keys  
9 [JScript] public var Clear : Keys;

10  
11 *Description*

12 The CLEAR key.

13 *u) ToString*

14  
15 [C#] public const Keys Control;  
16 [C++] public: const Keys Control;  
17 [VB] Public Const Control As Keys  
18 [JScript] public var Control : Keys;

19  
20 *Description*

21 The CTRL modifier key.

22 *v) ToString*

23  
24 [C#] public const Keys ControlKey;  
25 [C++] public: const Keys ControlKey;



|   |           |        |       |            |    |       |
|---|-----------|--------|-------|------------|----|-------|
| 1 | [VB]      | Public | Const | ControlKey | As | Keys  |
| 2 | [JScript] | public | var   | ControlKey | :  | Keys; |

3

4 *Description*

5 The CTRL key.

6       w)    ***ToString***

|    |           |         |       |        |    |         |
|----|-----------|---------|-------|--------|----|---------|
| 8  | [C#]      | public  | const | Keys   |    | Crssel; |
| 9  | [C++]     | public: | const | Keys   |    | Crssel; |
| 10 | [VB]      | Public  | Const | Crssel | As | Keys    |
| 11 | [JScript] | public  | var   | Crssel | :  | Keys;   |

12

13 *Description*

14 The CRSEL key.

15       x)    ***ToString***

|    |           |         |       |      |    |       |
|----|-----------|---------|-------|------|----|-------|
| 17 | [C#]      | public  | const | Keys |    | D;    |
| 18 | [C++]     | public: | const | Keys |    | D;    |
| 19 | [VB]      | Public  | Const | D    | As | Keys  |
| 20 | [JScript] | public  | var   | D    | :  | Keys; |

21

22 *Description*

23 The D key.

24

25

y) *ToString*

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D0;   |
| [C++]     | public: | const | Keys  | D0;   |
| [VB]      | Public  | Const | D0 As | Keys  |
| [JScript] | public  | var   | D0 :  | Keys; |

*Description*

The 0 key.

z) *ToString*

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D1;   |
| [C++]     | public: | const | Keys  | D1;   |
| [VB]      | Public  | Const | D1 As | Keys  |
| [JScript] | public  | var   | D1 :  | Keys; |

*Description*

The 1 key.

aa) *ToString*

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D2;   |
| [C++]     | public: | const | Keys  | D2;   |
| [VB]      | Public  | Const | D2 As | Keys  |
| [JScript] | public  | var   | D2 :  | Keys; |

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The 2 key.

*bb) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | D3;     |
| [C++]     | public: | const | Keys | D3;     |
| [VB]      | Public  | Const | D3   | As Keys |
| [JScript] | public  | var   | D3   | : Keys; |

*Description*

The 3 key.

*cc) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | D4;     |
| [C++]     | public: | const | Keys | D4;     |
| [VB]      | Public  | Const | D4   | As Keys |
| [JScript] | public  | var   | D4   | : Keys; |

*Description*

The 4 key.

*dd) ToString*

|       |         |       |      |     |
|-------|---------|-------|------|-----|
| [C#]  | public  | const | Keys | D5; |
| [C++] | public: | const | Keys | D5; |

|           |        |       |    |    |       |
|-----------|--------|-------|----|----|-------|
| [VB]      | Public | Const | D5 | As | Keys  |
| [JScript] | public | var   | D5 | :  | Keys; |

*Description*

The 5 key.

*ee) ToString*

|      |        |       |  |      |     |
|------|--------|-------|--|------|-----|
| [C#] | public | const |  | Keys | D6; |
|------|--------|-------|--|------|-----|

|       |         |       |  |      |     |
|-------|---------|-------|--|------|-----|
| [C++] | public: | const |  | Keys | D6; |
|-------|---------|-------|--|------|-----|

|      |        |       |    |    |      |
|------|--------|-------|----|----|------|
| [VB] | Public | Const | D6 | As | Keys |
|------|--------|-------|----|----|------|

|           |        |     |    |   |       |
|-----------|--------|-----|----|---|-------|
| [JScript] | public | var | D6 | : | Keys; |
|-----------|--------|-----|----|---|-------|

*Description*

The 6 key.

*ff) ToString*

|      |        |       |  |      |     |
|------|--------|-------|--|------|-----|
| [C#] | public | const |  | Keys | D7; |
|------|--------|-------|--|------|-----|

|       |         |       |  |      |     |
|-------|---------|-------|--|------|-----|
| [C++] | public: | const |  | Keys | D7; |
|-------|---------|-------|--|------|-----|

|      |        |       |    |    |      |
|------|--------|-------|----|----|------|
| [VB] | Public | Const | D7 | As | Keys |
|------|--------|-------|----|----|------|

|           |        |     |    |   |       |
|-----------|--------|-----|----|---|-------|
| [JScript] | public | var | D7 | : | Keys; |
|-----------|--------|-----|----|---|-------|

*Description*

The 7 key.

**gg) ToString**

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D8;   |
| [C++]     | public: | const | Keys  | D8;   |
| [VB]      | Public  | Const | D8 As | Keys  |
| [JScript] | public  | var   | D8 :  | Keys; |

*Description*

The 8 key.

**hh) ToString**

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | D9;   |
| [C++]     | public: | const | Keys  | D9;   |
| [VB]      | Public  | Const | D9 As | Keys  |
| [JScript] | public  | var   | D9 :  | Keys; |

*Description*

The 9 key.

**ii) ToString**

|           |         |       |            |          |
|-----------|---------|-------|------------|----------|
| [C#]      | public  | const | Keys       | Decimal; |
| [C++]     | public: | const | Keys       | Decimal; |
| [VB]      | Public  | Const | Decimal As | Keys     |
| [JScript] | public  | var   | Decimal :  | Keys;    |

*Description*

The Decimal key.

*jj) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Delete; |
| [C++]     | public: | const | Keys   | Delete; |
| [VB]      | Public  | Const | Delete | As Keys |
| [JScript] | public  | var   | Delete | : Keys; |

*Description*

The DEL key.

*kk) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Divide; |
| [C++]     | public: | const | Keys   | Divide; |
| [VB]      | Public  | Const | Divide | As Keys |
| [JScript] | public  | var   | Divide | : Keys; |

*Description*

The Divide key.

*ll) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Down; |
| [C++] | public: | const | Keys | Down; |

|    |                     |         |       |      |      |       |
|----|---------------------|---------|-------|------|------|-------|
| 1  | [VB]                | Public  | Const | Down | As   | Keys  |
| 2  | [JScript]           | public  | var   | Down | :    | Keys; |
| 3  |                     |         |       |      |      |       |
| 4  | <i>Description</i>  |         |       |      |      |       |
| 5  | The DOWN ARROW key. |         |       |      |      |       |
| 6  | <i>mm) ToString</i> |         |       |      |      |       |
| 7  |                     |         |       |      |      |       |
| 8  | [C#]                | public  | const |      | Keys | E;    |
| 9  | [C++]               | public: | const |      | Keys | E;    |
| 10 | [VB]                | Public  | Const | E    | As   | Keys  |
| 11 | [JScript]           | public  | var   | E    | :    | Keys; |
| 12 |                     |         |       |      |      |       |
| 13 | <i>Description</i>  |         |       |      |      |       |
| 14 | The E key.          |         |       |      |      |       |
| 15 | <i>nn) ToString</i> |         |       |      |      |       |
| 16 |                     |         |       |      |      |       |
| 17 | [C#]                | public  | const |      | Keys | End;  |
| 18 | [C++]               | public: | const |      | Keys | End;  |
| 19 | [VB]                | Public  | Const | End  | As   | Keys  |
| 20 | [JScript]           | public  | var   | End  | :    | Keys; |
| 21 |                     |         |       |      |      |       |
| 22 | <i>Description</i>  |         |       |      |      |       |
| 23 | The END key.        |         |       |      |      |       |
| 24 |                     |         |       |      |      |       |
| 25 |                     |         |       |      |      |       |

**oo) ToString**

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Enter;  |
| [C++]     | public: | const | Keys  | Enter;  |
| [VB]      | Public  | Const | Enter | As Keys |
| [JScript] | public  | var   | Enter | : Keys; |

*Description*

The ENTER key.

**pp) ToString**

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | EraseEof; |
| [C++]     | public: | const | Keys     | EraseEof; |
| [VB]      | Public  | Const | EraseEof | As Keys   |
| [JScript] | public  | var   | EraseEof | : Keys;   |

*Description*

The ERASE EOF key.

**qq) ToString**

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Escape; |
| [C++]     | public: | const | Keys   | Escape; |
| [VB]      | Public  | Const | Escape | As Keys |
| [JScript] | public  | var   | Escape | : Keys; |



*Description*

The ESC key.

*rr) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | Execute; |
| [C++]     | public: | const | Keys    | Execute; |
| [VB]      | Public  | Const | Execute | As Keys  |
| [JScript] | public  | var   | Execute | : Keys;  |

*Description*

The EXECUTE key.

*ss) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Exsel;  |
| [C++]     | public: | const | Keys  | Exsel;  |
| [VB]      | Public  | Const | Exsel | As Keys |
| [JScript] | public  | var   | Exsel | : Keys; |

*Description*

The EXSEL key.

*tt) ToString*

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C#]  | public  | const | Keys | F; |
| [C++] | public: | const | Keys | F; |

|    |                     |         |       |     |      |       |
|----|---------------------|---------|-------|-----|------|-------|
| 1  | [VB]                | Public  | Const | F   | As   | Keys  |
| 2  | [JScript]           | public  | var   | F   | :    | Keys; |
| 3  |                     |         |       |     |      |       |
| 4  | <i>Description</i>  |         |       |     |      |       |
| 5  | The F key.          |         |       |     |      |       |
| 6  | <i>uu) ToString</i> |         |       |     |      |       |
| 7  |                     |         |       |     |      |       |
| 8  | [C#]                | public  | const |     | Keys | F1;   |
| 9  | [C++]               | public: | const |     | Keys | F1;   |
| 10 | [VB]                | Public  | Const | F1  | As   | Keys  |
| 11 | [JScript]           | public  | var   | F1  | :    | Keys; |
| 12 |                     |         |       |     |      |       |
| 13 | <i>Description</i>  |         |       |     |      |       |
| 14 | The F1 key.         |         |       |     |      |       |
| 15 | <i>vv) ToString</i> |         |       |     |      |       |
| 16 |                     |         |       |     |      |       |
| 17 | [C#]                | public  | const |     | Keys | F10;  |
| 18 | [C++]               | public: | const |     | Keys | F10;  |
| 19 | [VB]                | Public  | Const | F10 | As   | Keys  |
| 20 | [JScript]           | public  | var   | F10 | :    | Keys; |
| 21 |                     |         |       |     |      |       |
| 22 | <i>Description</i>  |         |       |     |      |       |
| 23 | The F10 key.        |         |       |     |      |       |
| 24 |                     |         |       |     |      |       |
| 25 |                     |         |       |     |      |       |

*ww) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F11;  |
| [C++]     | public: | const | Keys   | F11;  |
| [VB]      | Public  | Const | F11 As | Keys  |
| [JScript] | public  | var   | F11 :  | Keys; |

*Description*

The F11 key.

*xx) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F12;  |
| [C++]     | public: | const | Keys   | F12;  |
| [VB]      | Public  | Const | F12 As | Keys  |
| [JScript] | public  | var   | F12 :  | Keys; |

*Description*

The F12 key.

*yy) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F13;  |
| [C++]     | public: | const | Keys   | F13;  |
| [VB]      | Public  | Const | F13 As | Keys  |
| [JScript] | public  | var   | F13 :  | Keys; |

*Description*

The F13 key.

*zz) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F14;    |
| [C++]     | public: | const | Keys | F14;    |
| [VB]      | Public  | Const | F14  | As Keys |
| [JScript] | public  | var   | F14  | : Keys; |

*Description*

The F14 key.

*aaa) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F15;    |
| [C++]     | public: | const | Keys | F15;    |
| [VB]      | Public  | Const | F15  | As Keys |
| [JScript] | public  | var   | F15  | : Keys; |

*Description*

The F15 key.

*bbb) ToString*

|       |         |       |      |      |
|-------|---------|-------|------|------|
| [C#]  | public  | const | Keys | F16; |
| [C++] | public: | const | Keys | F16; |

|    |                      |         |       |     |      |       |
|----|----------------------|---------|-------|-----|------|-------|
| 1  | [VB]                 | Public  | Const | F16 | As   | Keys  |
| 2  | [JScript]            | public  | var   | F16 | :    | Keys; |
| 3  |                      |         |       |     |      |       |
| 4  | <i>Description</i>   |         |       |     |      |       |
| 5  | The F16 key.         |         |       |     |      |       |
| 6  | <i>ccc) ToString</i> |         |       |     |      |       |
| 7  |                      |         |       |     |      |       |
| 8  | [C#]                 | public  | const |     | Keys | F17;  |
| 9  | [C++]                | public: | const |     | Keys | F17;  |
| 10 | [VB]                 | Public  | Const | F17 | As   | Keys  |
| 11 | [JScript]            | public  | var   | F17 | :    | Keys; |
| 12 |                      |         |       |     |      |       |
| 13 | <i>Description</i>   |         |       |     |      |       |
| 14 | The F17 key.         |         |       |     |      |       |
| 15 | <i>ddd) ToString</i> |         |       |     |      |       |
| 16 |                      |         |       |     |      |       |
| 17 | [C#]                 | public  | const |     | Keys | F18;  |
| 18 | [C++]                | public: | const |     | Keys | F18;  |
| 19 | [VB]                 | Public  | Const | F18 | As   | Keys  |
| 20 | [JScript]            | public  | var   | F18 | :    | Keys; |
| 21 |                      |         |       |     |      |       |
| 22 | <i>Description</i>   |         |       |     |      |       |
| 23 | The F18 key.         |         |       |     |      |       |
| 24 |                      |         |       |     |      |       |
| 25 |                      |         |       |     |      |       |

*eee) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F19;  |
| [C++]     | public: | const | Keys   | F19;  |
| [VB]      | Public  | Const | F19 As | Keys  |
| [JScript] | public  | var   | F19 :  | Keys; |

*Description*

The F19 key.

*fff) ToString*

|           |         |       |       |       |
|-----------|---------|-------|-------|-------|
| [C#]      | public  | const | Keys  | F2;   |
| [C++]     | public: | const | Keys  | F2;   |
| [VB]      | Public  | Const | F2 As | Keys  |
| [JScript] | public  | var   | F2 :  | Keys; |

*Description*

The F2 key.

*ggg) ToString*

|           |         |       |        |       |
|-----------|---------|-------|--------|-------|
| [C#]      | public  | const | Keys   | F20;  |
| [C++]     | public: | const | Keys   | F20;  |
| [VB]      | Public  | Const | F20 As | Keys  |
| [JScript] | public  | var   | F20 :  | Keys; |

*Description*

The F20 key.

*hhh) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F21;    |
| [C++]     | public: | const | Keys | F21;    |
| [VB]      | Public  | Const | F21  | As Keys |
| [JScript] | public  | var   | F21  | : Keys; |

*Description*

The F21 key.

*iii) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F22;    |
| [C++]     | public: | const | Keys | F22;    |
| [VB]      | Public  | Const | F22  | As Keys |
| [JScript] | public  | var   | F22  | : Keys; |

*Description*

The F22 key.

*jjj) ToString*

|       |         |       |      |      |
|-------|---------|-------|------|------|
| [C#]  | public  | const | Keys | F23; |
| [C++] | public: | const | Keys | F23; |

|    |                      |         |       |     |      |       |
|----|----------------------|---------|-------|-----|------|-------|
| 1  | [VB]                 | Public  | Const | F23 | As   | Keys  |
| 2  | [JScript]            | public  | var   | F23 | :    | Keys; |
| 3  |                      |         |       |     |      |       |
| 4  | <i>Description</i>   |         |       |     |      |       |
| 5  | The F23 key.         |         |       |     |      |       |
| 6  | <i>kkk) ToString</i> |         |       |     |      |       |
| 7  |                      |         |       |     |      |       |
| 8  | [C#]                 | public  | const |     | Keys | F24;  |
| 9  | [C++]                | public: | const |     | Keys | F24;  |
| 10 | [VB]                 | Public  | Const | F24 | As   | Keys  |
| 11 | [JScript]            | public  | var   | F24 | :    | Keys; |
| 12 |                      |         |       |     |      |       |
| 13 | <i>Description</i>   |         |       |     |      |       |
| 14 | The F24 key.         |         |       |     |      |       |
| 15 | <i>lll) ToString</i> |         |       |     |      |       |
| 16 |                      |         |       |     |      |       |
| 17 | [C#]                 | public  | const |     | Keys | F3;   |
| 18 | [C++]                | public: | const |     | Keys | F3;   |
| 19 | [VB]                 | Public  | Const | F3  | As   | Keys  |
| 20 | [JScript]            | public  | var   | F3  | :    | Keys; |
| 21 |                      |         |       |     |      |       |
| 22 | <i>Description</i>   |         |       |     |      |       |
| 23 | The F3 key.          |         |       |     |      |       |
| 24 |                      |         |       |     |      |       |
| 25 |                      |         |       |     |      |       |



*mmm) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F4;     |
| [C++]     | public: | const | Keys | F4;     |
| [VB]      | Public  | Const | F4   | As Keys |
| [JScript] | public  | var   | F4   | : Keys; |

*Description*

The F4 key.

*nnn) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F5;     |
| [C++]     | public: | const | Keys | F5;     |
| [VB]      | Public  | Const | F5   | As Keys |
| [JScript] | public  | var   | F5   | : Keys; |

*Description*

The F5 key.

*ooo) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F6;     |
| [C++]     | public: | const | Keys | F6;     |
| [VB]      | Public  | Const | F6   | As Keys |
| [JScript] | public  | var   | F6   | : Keys; |

11/03/2004 10:00:00 AM

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

*Description*

The F6 key.

*ppp) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F7;     |
| [C++]     | public: | const | Keys | F7;     |
| [VB]      | Public  | Const | F7   | As Keys |
| [JScript] | public  | var   | F7   | : Keys; |

*Description*

The F7 key.

*qqq) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | F8;     |
| [C++]     | public: | const | Keys | F8;     |
| [VB]      | Public  | Const | F8   | As Keys |
| [JScript] | public  | var   | F8   | : Keys; |

*Description*

The F8 key.

*rrr) ToString*

|       |         |       |      |     |
|-------|---------|-------|------|-----|
| [C#]  | public  | const | Keys | F9; |
| [C++] | public: | const | Keys | F9; |

|   |           |        |       |    |    |       |
|---|-----------|--------|-------|----|----|-------|
| 1 | [VB]      | Public | Const | F9 | As | Keys  |
| 2 | [JScript] | public | var   | F9 | :  | Keys; |

3

4 *Description*

5 The F9 key.

6 *sss) ToString*

7

|   |      |        |       |      |            |
|---|------|--------|-------|------|------------|
| 8 | [C#] | public | const | Keys | FinalMode; |
|---|------|--------|-------|------|------------|

|   |       |         |       |      |            |
|---|-------|---------|-------|------|------------|
| 9 | [C++] | public: | const | Keys | FinalMode; |
|---|-------|---------|-------|------|------------|

|    |      |        |       |           |    |      |
|----|------|--------|-------|-----------|----|------|
| 10 | [VB] | Public | Const | FinalMode | As | Keys |
|----|------|--------|-------|-----------|----|------|

|    |           |        |     |           |   |       |
|----|-----------|--------|-----|-----------|---|-------|
| 11 | [JScript] | public | var | FinalMode | : | Keys; |
|----|-----------|--------|-----|-----------|---|-------|

12

13 *Description*

14 The IME final mode key.

15 *ttt) ToString*

16

|    |      |        |       |      |    |
|----|------|--------|-------|------|----|
| 17 | [C#] | public | const | Keys | G; |
|----|------|--------|-------|------|----|

|    |       |         |       |      |    |
|----|-------|---------|-------|------|----|
| 18 | [C++] | public: | const | Keys | G; |
|----|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 19 | [VB] | Public | Const | G | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 20 | [JScript] | public | var | G | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

21

22 *Description*

23 The G key.

24

25

*uuu) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | H;      |
| [C++]     | public: | const | Keys | H;      |
| [VB]      | Public  | Const | H    | As Keys |
| [JScript] | public  | var   | H    | : Keys; |

*Description*

The H key.

*vvv) ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | HanguelMode; |
| [C++]     | public: | const | Keys        | HanguelMode; |
| [VB]      | Public  | Const | HanguelMode | As Keys      |
| [JScript] | public  | var   | HanguelMode | : Keys;      |

*Description*

The IME Hanguel mode key. (maintained for compatibility; use **HangulMode** )  
The IME Hanguel mode key.

*www) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | HangulMode; |
| [C++]     | public: | const | Keys       | HangulMode; |
| [VB]      | Public  | Const | HangulMode | As Keys     |
| [JScript] | public  | var   | HangulMode | : Keys;     |

*Description*

The IME Hangul mode key.

*xxx) ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | HanjaMode; |
| [C++]     | public: | const | Keys      | HanjaMode; |
| [VB]      | Public  | Const | HanjaMode | As Keys    |
| [JScript] | public  | var   | HanjaMode | : Keys;    |

*Description*

The IME Hanja mode key.

*yyy) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Help;   |
| [C++]     | public: | const | Keys | Help;   |
| [VB]      | Public  | Const | Help | As Keys |
| [JScript] | public  | var   | Help | : Keys; |

*Description*

The HELP key.

*zzz) ToString*

|       |         |       |      |       |
|-------|---------|-------|------|-------|
| [C#]  | public  | const | Keys | Home; |
| [C++] | public: | const | Keys | Home; |

|    |                       |         |       |           |            |       |
|----|-----------------------|---------|-------|-----------|------------|-------|
| 1  | [VB]                  | Public  | Const | Home      | As         | Keys  |
| 2  | [JScript]             | public  | var   | Home      | :          | Keys; |
| 3  |                       |         |       |           |            |       |
| 4  | <i>Description</i>    |         |       |           |            |       |
| 5  | The HOME key.         |         |       |           |            |       |
| 6  | <b>aaaa) ToString</b> |         |       |           |            |       |
| 7  |                       |         |       |           |            |       |
| 8  | [C#]                  | public  | const | Keys      | I;         |       |
| 9  | [C++]                 | public: | const | Keys      | I;         |       |
| 10 | [VB]                  | Public  | Const | I         | As         | Keys  |
| 11 | [JScript]             | public  | var   | I         | :          | Keys; |
| 12 |                       |         |       |           |            |       |
| 13 | <i>Description</i>    |         |       |           |            |       |
| 14 | The I key.            |         |       |           |            |       |
| 15 | <b>bbbb) ToString</b> |         |       |           |            |       |
| 16 |                       |         |       |           |            |       |
| 17 | [C#]                  | public  | const | Keys      | IMEAccept; |       |
| 18 | [C++]                 | public: | const | Keys      | IMEAccept; |       |
| 19 | [VB]                  | Public  | Const | IMEAccept | As         | Keys  |
| 20 | [JScript]             | public  | var   | IMEAccept | :          | Keys; |
| 21 |                       |         |       |           |            |       |
| 22 | <i>Description</i>    |         |       |           |            |       |
| 23 | The IME Accept key.   |         |       |           |            |       |
| 24 |                       |         |       |           |            |       |
| 25 |                       |         |       |           |            |       |

*cccc) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | IMEConvert; |
| [C++]     | public: | const | Keys       | IMEConvert; |
| [VB]      | Public  | Const | IMEConvert | As Keys     |
| [JScript] | public  | var   | IMEConvert | : Keys;     |

*Description*

The IME Convert key.

*dddd) ToString*

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C#]      | public  | const | Keys          | IMEModeChange; |
| [C++]     | public: | const | Keys          | IMEModeChange; |
| [VB]      | Public  | Const | IMEModeChange | As Keys        |
| [JScript] | public  | var   | IMEModeChange | : Keys;        |

*Description*

The IME Mode Change key.

*eeee) ToString*

|           |         |       |               |                |
|-----------|---------|-------|---------------|----------------|
| [C#]      | public  | const | Keys          | IMENonconvert; |
| [C++]     | public: | const | Keys          | IMENonconvert; |
| [VB]      | Public  | Const | IMENonconvert | As Keys        |
| [JScript] | public  | var   | IMENonconvert | : Keys;        |

*Description*

The IME Nonconvert key.

*ffff) ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Insert; |
| [C++]     | public: | const | Keys   | Insert; |
| [VB]      | Public  | Const | Insert | As Keys |
| [JScript] | public  | var   | Insert | : Keys; |

*Description*

The INS key.

*gggg) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | J;      |
| [C++]     | public: | const | Keys | J;      |
| [VB]      | Public  | Const | J    | As Keys |
| [JScript] | public  | var   | J    | : Keys; |

*Description*

The J key.

*hhhh) ToString*

|       |         |       |      |            |
|-------|---------|-------|------|------------|
| [C#]  | public  | const | Keys | JunjaMode; |
| [C++] | public: | const | Keys | JunjaMode; |



|   |           |        |       |           |    |       |
|---|-----------|--------|-------|-----------|----|-------|
| 1 | [VB]      | Public | Const | JunjaMode | As | Keys  |
| 2 | [JScript] | public | var   | JunjaMode | :  | Keys; |

3

4 *Description*

5 The IME Junja mode key.

6 *iii) ToString*

7

|   |      |        |       |      |    |
|---|------|--------|-------|------|----|
| 8 | [C#] | public | const | Keys | K; |
|---|------|--------|-------|------|----|

|   |       |         |       |      |    |
|---|-------|---------|-------|------|----|
| 9 | [C++] | public: | const | Keys | K; |
|---|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 10 | [VB] | Public | Const | K | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 11 | [JScript] | public | var | K | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

12

13 *Description*

14 The K key.

15 *jjij) ToString*

16

|    |      |        |       |      |           |
|----|------|--------|-------|------|-----------|
| 17 | [C#] | public | const | Keys | KanaMode; |
|----|------|--------|-------|------|-----------|

|    |       |         |       |      |           |
|----|-------|---------|-------|------|-----------|
| 18 | [C++] | public: | const | Keys | KanaMode; |
|----|-------|---------|-------|------|-----------|

|    |      |        |       |          |    |      |
|----|------|--------|-------|----------|----|------|
| 19 | [VB] | Public | Const | KanaMode | As | Keys |
|----|------|--------|-------|----------|----|------|

|    |           |        |     |          |   |       |
|----|-----------|--------|-----|----------|---|-------|
| 20 | [JScript] | public | var | KanaMode | : | Keys; |
|----|-----------|--------|-----|----------|---|-------|

21

22 *Description*

23 The IME Kana mode key.

24

25

**kkkk) ToString**

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | KanjiMode; |
| [C++]     | public: | const | Keys      | KanjiMode; |
| [VB]      | Public  | Const | KanjiMode | As Keys    |
| [JScript] | public  | var   | KanjiMode | : Keys;    |

**Description**

The IME Kanji mode key.

**llll) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | KeyCode; |
| [C++]     | public: | const | Keys    | KeyCode; |
| [VB]      | Public  | Const | KeyCode | As Keys  |
| [JScript] | public  | var   | KeyCode | : Keys;  |

**Description**

The bitmask to extract a key code from a key value.

**mmmm)ToString**

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | L;      |
| [C++]     | public: | const | Keys | L;      |
| [VB]      | Public  | Const | L    | As Keys |
| [JScript] | public  | var   | L    | : Keys; |

*Description*

The L key.

*nnnn) ToString*

[C#]            public            const            Keys            LaunchApplication1;

[C++]           public:            const            Keys            LaunchApplication1;

[VB]            Public            Const            LaunchApplication1            As            Keys

[JScript]       public            var            LaunchApplication1            :            Keys;

*Description*

The Start Application one key (Windows 2000 or later).

*oooo) ToString*

[C#]            public            const            Keys            LaunchApplication2;

[C++]           public:            const            Keys            LaunchApplication2;

[VB]            Public            Const            LaunchApplication2            As            Keys

[JScript]       public            var            LaunchApplication2            :            Keys;

*Description*

The Start Application two key (Windows 2000 or later).

*pppp) ToString*

[C#]            public            const            Keys            LaunchMail;

[C++]           public:            const            Keys            LaunchMail;

|           |        |       |            |    |       |
|-----------|--------|-------|------------|----|-------|
| [VB]      | Public | Const | LaunchMail | As | Keys  |
| [JScript] | public | var   | LaunchMail | :  | Keys; |

*Description*

The Launch Mail key (Windows 2000 or later).

*qqqq) ToString*

|      |        |       |      |          |
|------|--------|-------|------|----------|
| [C#] | public | const | Keys | LButton; |
|------|--------|-------|------|----------|

|       |         |       |      |          |
|-------|---------|-------|------|----------|
| [C++] | public: | const | Keys | LButton; |
|-------|---------|-------|------|----------|

|      |        |       |         |    |      |
|------|--------|-------|---------|----|------|
| [VB] | Public | Const | LButton | As | Keys |
|------|--------|-------|---------|----|------|

|           |        |     |         |   |       |
|-----------|--------|-----|---------|---|-------|
| [JScript] | public | var | LButton | : | Keys; |
|-----------|--------|-----|---------|---|-------|

*Description*

The left mouse button.

*rrrr) ToString*

|      |        |       |      |              |
|------|--------|-------|------|--------------|
| [C#] | public | const | Keys | LControlKey; |
|------|--------|-------|------|--------------|

|       |         |       |      |              |
|-------|---------|-------|------|--------------|
| [C++] | public: | const | Keys | LControlKey; |
|-------|---------|-------|------|--------------|

|      |        |       |             |    |      |
|------|--------|-------|-------------|----|------|
| [VB] | Public | Const | LControlKey | As | Keys |
|------|--------|-------|-------------|----|------|

|           |        |     |             |   |       |
|-----------|--------|-----|-------------|---|-------|
| [JScript] | public | var | LControlKey | : | Keys; |
|-----------|--------|-----|-------------|---|-------|

*Description*

The left CTRL key.

*ssss) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Left;   |
| [C++]     | public: | const | Keys | Left;   |
| [VB]      | Public  | Const | Left | As Keys |
| [JScript] | public  | var   | Left | : Keys; |

*Description*

The LEFT ARROW key.

*tttt) ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | LineFeed; |
| [C++]     | public: | const | Keys     | LineFeed; |
| [VB]      | Public  | Const | LineFeed | As Keys   |
| [JScript] | public  | var   | LineFeed | : Keys;   |

*Description*

The LINEFEED key.

*uuuu) ToString*

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | LMenu;  |
| [C++]     | public: | const | Keys  | LMenu;  |
| [VB]      | Public  | Const | LMenu | As Keys |
| [JScript] | public  | var   | LMenu | : Keys; |

1  
2 *Description*

3 The left ALT key.

4 *vvvv) ToString*

5  
6 [C#] public const Keys LShiftKey;

7 [C++] public: const Keys LShiftKey;

8 [VB] Public Const LShiftKey As Keys

9 [JScript] public var LShiftKey : Keys;

10  
11 *Description*

12 The left SHIFT key.

13 *www)ToString*

14  
15 [C#] public const Keys LWin;

16 [C++] public: const Keys LWin;

17 [VB] Public Const LWin As Keys

18 [JScript] public var LWin : Keys;

19  
20 *Description*

21 The left Windows logo key (Microsoft Natural Keyboard).

22 *xxxx) ToString*

23  
24 [C#] public const Keys M;

25 [C++] public: const Keys M;

|           |        |       |   |    |       |
|-----------|--------|-------|---|----|-------|
| [VB]      | Public | Const | M | As | Keys  |
| [JScript] | public | var   | M | :  | Keys; |

#### Description

The M key.

yyyy) *ToString*

|      |        |       |      |          |
|------|--------|-------|------|----------|
| [C#] | public | const | Keys | MButton; |
|------|--------|-------|------|----------|

|       |         |       |      |          |
|-------|---------|-------|------|----------|
| [C++] | public: | const | Keys | MButton; |
|-------|---------|-------|------|----------|

|      |        |       |         |    |      |
|------|--------|-------|---------|----|------|
| [VB] | Public | Const | MButton | As | Keys |
|------|--------|-------|---------|----|------|

|           |        |     |         |   |       |
|-----------|--------|-----|---------|---|-------|
| [JScript] | public | var | MButton | : | Keys; |
|-----------|--------|-----|---------|---|-------|

#### Description

The middle mouse button (three-button mouse).

zzzz) *ToString*

|      |        |       |      |                 |
|------|--------|-------|------|-----------------|
| [C#] | public | const | Keys | MediaNextTrack; |
|------|--------|-------|------|-----------------|

|       |         |       |      |                 |
|-------|---------|-------|------|-----------------|
| [C++] | public: | const | Keys | MediaNextTrack; |
|-------|---------|-------|------|-----------------|

|      |        |       |                |    |      |
|------|--------|-------|----------------|----|------|
| [VB] | Public | Const | MediaNextTrack | As | Keys |
|------|--------|-------|----------------|----|------|

|           |        |     |                |   |       |
|-----------|--------|-----|----------------|---|-------|
| [JScript] | public | var | MediaNextTrack | : | Keys; |
|-----------|--------|-----|----------------|---|-------|

#### Description

The Media Next Track key (Windows 2000 or later).

***aaaaa) ToString***

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | Keys           | MediaPlayPause; |
| [C++]     | public: | const | Keys           | MediaPlayPause; |
| [VB]      | Public  | Const | MediaPlayPause | As Keys         |
| [JScript] | public  | var   | MediaPlayPause | : Keys;         |

***Description***

The Media Play Pause key (Windows 2000 or later).

***bbbb) ToString***

|           |         |       |                    |                     |
|-----------|---------|-------|--------------------|---------------------|
| [C#]      | public  | const | Keys               | MediaPreviousTrack; |
| [C++]     | public: | const | Keys               | MediaPreviousTrack; |
| [VB]      | Public  | Const | MediaPreviousTrack | As Keys             |
| [JScript] | public  | var   | MediaPreviousTrack | : Keys;             |

***Description***

The Media Previous Track key (Windows 2000 or later).

***cccc) ToString***

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | MediaStop; |
| [C++]     | public: | const | Keys      | MediaStop; |
| [VB]      | Public  | Const | MediaStop | As Keys    |
| [JScript] | public  | var   | MediaStop | : Keys;    |



*Description*

The Media Stop key (Windows 2000 or later).

*dddd) ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Menu;   |
| [C++]     | public: | const | Keys | Menu;   |
| [VB]      | Public  | Const | Menu | As Keys |
| [JScript] | public  | var   | Menu | : Keys; |

*Description*

The ALT key.

*eeee) ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | Modifiers; |
| [C++]     | public: | const | Keys      | Modifiers; |
| [VB]      | Public  | Const | Modifiers | As Keys    |
| [JScript] | public  | var   | Modifiers | : Keys;    |

*Description*

The bitmask to extract modifiers from a key value.

*ffff) ToString*

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C#]  | public  | const | Keys | Multiply; |
| [C++] | public: | const | Keys | Multiply; |

|    |                        |         |       |          |    |       |
|----|------------------------|---------|-------|----------|----|-------|
| 1  | [VB]                   | Public  | Const | Multiply | As | Keys  |
| 2  | [JScript]              | public  | var   | Multiply | :  | Keys; |
| 3  |                        |         |       |          |    |       |
| 4  | <i>Description</i>     |         |       |          |    |       |
| 5  | The Multiply key.      |         |       |          |    |       |
| 6  | <b>ggggg) ToString</b> |         |       |          |    |       |
| 7  |                        |         |       |          |    |       |
| 8  | [C#]                   | public  | const | Keys     |    | N;    |
| 9  | [C++]                  | public: | const | Keys     |    | N;    |
| 10 | [VB]                   | Public  | Const | N        | As | Keys  |
| 11 | [JScript]              | public  | var   | N        | :  | Keys; |
| 12 |                        |         |       |          |    |       |
| 13 | <i>Description</i>     |         |       |          |    |       |
| 14 | The N key.             |         |       |          |    |       |
| 15 | <b>hhhhh) ToString</b> |         |       |          |    |       |
| 16 |                        |         |       |          |    |       |
| 17 | [C#]                   | public  | const | Keys     |    | Next; |
| 18 | [C++]                  | public: | const | Keys     |    | Next; |
| 19 | [VB]                   | Public  | Const | Next     | As | Keys  |
| 20 | [JScript]              | public  | var   | Next     | :  | Keys; |
| 21 |                        |         |       |          |    |       |
| 22 | <i>Description</i>     |         |       |          |    |       |
| 23 | The PAGE DOWN key.     |         |       |          |    |       |
| 24 |                        |         |       |          |    |       |
| 25 |                        |         |       |          |    |       |

iiii) ToString

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | NoName; |
| [C++]     | public: | const | Keys   | NoName; |
| [VB]      | Public  | Const | NoName | As Keys |
| [JScript] | public  | var   | NoName | : Keys; |

Description

A constant reserved for future use.

jjjj) ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | None;   |
| [C++]     | public: | const | Keys | None;   |
| [VB]      | Public  | Const | None | As Keys |
| [JScript] | public  | var   | None | : Keys; |

Description

No key pressed.

kkkkk) ToString

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumLock; |
| [C++]     | public: | const | Keys    | NumLock; |
| [VB]      | Public  | Const | NumLock | As Keys  |
| [JScript] | public  | var   | NumLock | : Keys;  |

*Description*

The NUM LOCK key.

*IIII) ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad0; |
| [C++]     | public: | const | Keys    | NumPad0; |
| [VB]      | Public  | Const | NumPad0 | As Keys  |
| [JScript] | public  | var   | NumPad0 | : Keys;  |

*Description*

The 0 key on the numeric keypad.

*mmmmm)ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad1; |
| [C++]     | public: | const | Keys    | NumPad1; |
| [VB]      | Public  | Const | NumPad1 | As Keys  |
| [JScript] | public  | var   | NumPad1 | : Keys;  |

*Description*

The 1 key on the numeric keypad.

*nnnnn)ToString*

|       |         |       |      |          |
|-------|---------|-------|------|----------|
| [C#]  | public  | const | Keys | NumPad2; |
| [C++] | public: | const | Keys | NumPad2; |

1 [VB] Public Const NumPad2 As Keys  
 2 [JScript] public var NumPad2 : Keys;

3  
 4 *Description*  
 5 The 2 key on the numeric keypad.

6 *ooooo) ToString*

7  
 8 [C#] public const Keys NumPad3;  
 9 [C++] public: const Keys NumPad3;  
 10 [VB] Public Const NumPad3 As Keys  
 11 [JScript] public var NumPad3 : Keys;

12  
 13 *Description*  
 14 The 3 key on the numeric keypad.

15 *ppppp) ToString*

16  
 17 [C#] public const Keys NumPad4;  
 18 [C++] public: const Keys NumPad4;  
 19 [VB] Public Const NumPad4 As Keys  
 20 [JScript] public var NumPad4 : Keys;

21  
 22 *Description*  
 23 The 4 key on the numeric keypad.

24  
 25

**qqqqq) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad5; |
| [C++]     | public: | const | Keys    | NumPad5; |
| [VB]      | Public  | Const | NumPad5 | As Keys  |
| [JScript] | public  | var   | NumPad5 | : Keys;  |

**Description**

The 5 key on the numeric keypad.

**rrrrr) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad6; |
| [C++]     | public: | const | Keys    | NumPad6; |
| [VB]      | Public  | Const | NumPad6 | As Keys  |
| [JScript] | public  | var   | NumPad6 | : Keys;  |

**Description**

The 6 key on the numeric keypad.

**sssss) ToString**

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | NumPad7; |
| [C++]     | public: | const | Keys    | NumPad7; |
| [VB]      | Public  | Const | NumPad7 | As Keys  |
| [JScript] | public  | var   | NumPad7 | : Keys;  |

1  
2 *Description*

3 The 7 key on the numeric keypad.

4 *ttttt) ToString*

5  
6 [C#] public const Keys NumPad8;

7 [C++] public: const Keys NumPad8;

8 [VB] Public Const NumPad8 As Keys

9 [JScript] public var NumPad8 : Keys;

10  
11 *Description*

12 The 8 key on the numeric keypad.

13 *uuuuu)ToString*

14  
15 [C#] public const Keys NumPad9;

16 [C++] public: const Keys NumPad9;

17 [VB] Public Const NumPad9 As Keys

18 [JScript] public var NumPad9 : Keys;

19  
20 *Description*

21 The 9 key on the numeric keypad.

22 *vvvvv) ToString*

23  
24 [C#] public const Keys O;

25 [C++] public: const Keys O;

|    |                                                                                            |         |       |              |    |               |
|----|--------------------------------------------------------------------------------------------|---------|-------|--------------|----|---------------|
| 1  | [VB]                                                                                       | Public  | Const | O            | As | Keys          |
| 2  | [JScript]                                                                                  | public  | var   | O            | :  | Keys;         |
| 3  |                                                                                            |         |       |              |    |               |
| 4  | <i>Description</i>                                                                         |         |       |              |    |               |
| 5  | The O key.                                                                                 |         |       |              |    |               |
| 6  | <i>wwwwww)ToString</i>                                                                     |         |       |              |    |               |
| 7  |                                                                                            |         |       |              |    |               |
| 8  | [C#]                                                                                       | public  | const | Keys         |    | Oem8;         |
| 9  | [C++]                                                                                      | public: | const | Keys         |    | Oem8;         |
| 10 | [VB]                                                                                       | Public  | Const | Oem8         | As | Keys          |
| 11 | [JScript]                                                                                  | public  | var   | Oem8         | :  | Keys;         |
| 12 |                                                                                            |         |       |              |    |               |
| 13 | <i>Description</i>                                                                         |         |       |              |    |               |
| 14 | OEM specific.                                                                              |         |       |              |    |               |
| 15 | <i>xxxxx) ToString</i>                                                                     |         |       |              |    |               |
| 16 |                                                                                            |         |       |              |    |               |
| 17 | [C#]                                                                                       | public  | const | Keys         |    | OemBackslash; |
| 18 | [C++]                                                                                      | public: | const | Keys         |    | OemBackslash; |
| 19 | [VB]                                                                                       | Public  | Const | OemBackslash | As | Keys          |
| 20 | [JScript]                                                                                  | public  | var   | OemBackslash | :  | Keys;         |
| 21 |                                                                                            |         |       |              |    |               |
| 22 | <i>Description</i>                                                                         |         |       |              |    |               |
| 23 | The OEM Angle bracket or Backslash key on the RT 102 key keyboard (Windows 2000 or later). |         |       |              |    |               |
| 24 |                                                                                            |         |       |              |    |               |
| 25 |                                                                                            |         |       |              |    |               |



yyyyy) ToString

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | OemClear; |
| [C++]     | public: | const | Keys     | OemClear; |
| [VB]      | Public  | Const | OemClear | As Keys   |
| [JScript] | public  | var   | OemClear | : Keys;   |

Description

The CLEAR key.

zzzzz) ToString

|           |         |       |                  |                   |
|-----------|---------|-------|------------------|-------------------|
| [C#]      | public  | const | Keys             | OemCloseBrackets; |
| [C++]     | public: | const | Keys             | OemCloseBrackets; |
| [VB]      | Public  | Const | OemCloseBrackets | As Keys           |
| [JScript] | public  | var   | OemCloseBrackets | : Keys;           |

Description

The OEM Close Bracket key on a US standard keyboard (Windows 2000 or later).

aaaaaa)ToString

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | Oemcomma; |
| [C++]     | public: | const | Keys     | Oemcomma; |
| [VB]      | Public  | Const | Oemcomma | As Keys   |
| [JScript] | public  | var   | Oemcomma | : Keys;   |

1  
2 *Description*

3 The OEM Comma key on any country/region keyboard (Windows 2000 or later).

4 *bbbbbb)ToString*

5  
6 [C#] public const Keys OemMinus;  
7 [C++] public: const Keys OemMinus;  
8 [VB] Public Const OemMinus As Keys  
9 [JScript] public var OemMinus : Keys;

10  
11 *Description*

12 The OEM Minus key on any country/region keyboard (Windows 2000 or later).

13 *cccccc)ToString*

14  
15 [C#] public const Keys OemOpenBrackets;  
16 [C++] public: const Keys OemOpenBrackets;  
17 [VB] Public Const OemOpenBrackets As Keys  
18 [JScript] public var OemOpenBrackets : Keys;

19  
20 *Description*

21 The OEM OpenBracket key on a US standard keyboard (Windows 2000 or later).

22 *dddddd)ToString*

23  
24 [C#] public const Keys OemPeriod;  
25 [C++] public: const Keys OemPeriod;

|           |        |       |           |    |       |
|-----------|--------|-------|-----------|----|-------|
| [VB]      | Public | Const | OemPeriod | As | Keys  |
| [JScript] | public | var   | OemPeriod | :  | Keys; |

*Description*

The OEM Period key on any country/region keyboard (Windows 2000 or later).

*eeeeee)ToString*

|           |         |       |         |          |       |
|-----------|---------|-------|---------|----------|-------|
| [C#]      | public  | const | Keys    | OemPipe; |       |
| [C++]     | public: | const | Keys    | OemPipe; |       |
| [VB]      | Public  | Const | OemPipe | As       | Keys  |
| [JScript] | public  | var   | OemPipe | :        | Keys; |

*Description*

The OEM Pipe key on a US standard keyboard (Windows 2000 or later).

*ffffff) ToString*

|           |         |       |         |          |       |
|-----------|---------|-------|---------|----------|-------|
| [C#]      | public  | const | Keys    | Oemplus; |       |
| [C++]     | public: | const | Keys    | Oemplus; |       |
| [VB]      | Public  | Const | Oemplus | As       | Keys  |
| [JScript] | public  | var   | Oemplus | :        | Keys; |

*Description*

The OEM Plus key on any country/region keyboard (Windows 2000 or later).

**gggggg)ToString**

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | OemQuestion; |
| [C++]     | public: | const | Keys        | OemQuestion; |
| [VB]      | Public  | Const | OemQuestion | As Keys      |
| [JScript] | public  | var   | OemQuestion | : Keys;      |

**Description**

The OEM Question Mark key on a US standard keyboard (Windows 2000 or later).

**hhhhh)ToString**

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | Keys      | OemQuotes; |
| [C++]     | public: | const | Keys      | OemQuotes; |
| [VB]      | Public  | Const | OemQuotes | As Keys    |
| [JScript] | public  | var   | OemQuotes | : Keys;    |

**Description**

The OEM Singled/Double quote key on a US standard keyboard (Windows 2000 or later).

**iiiiii) ToString**

|           |         |       |              |               |
|-----------|---------|-------|--------------|---------------|
| [C#]      | public  | const | Keys         | OemSemicolon; |
| [C++]     | public: | const | Keys         | OemSemicolon; |
| [VB]      | Public  | Const | OemSemicolon | As Keys       |
| [JScript] | public  | var   | OemSemicolon | : Keys;       |

1  
2 *Description*

3 The OEM Semicolon key on a US standard keyboard (Windows 2000 or later).

4 *jjjjj) ToString*

5  
6 [C#] public const Keys Oemtilde;

7 [C++] public: const Keys Oemtilde;

8 [VB] Public Const Oemtilde As Keys

9 [JScript] public var Oemtilde : Keys;

10  
11 *Description*

12 The OEM tilde key on a US standard keyboard (Windows 2000 or later).

13 *kkkkkk)ToString*

14  
15 [C#] public const Keys P;

16 [C++] public: const Keys P;

17 [VB] Public Const P As Keys

18 [JScript] public var P : Keys;

19  
20 *Description*

21 The P key.

22 *lllll) ToString*

23  
24 [C#] public const Keys Pal;

25 [C++] public: const Keys Pal;

|    |                        |         |       |          |    |           |
|----|------------------------|---------|-------|----------|----|-----------|
| 1  | [VB]                   | Public  | Const | Pa1      | As | Keys      |
| 2  | [JScript]              | public  | var   | Pa1      | :  | Keys;     |
| 3  |                        |         |       |          |    |           |
| 4  | <i>Description</i>     |         |       |          |    |           |
| 5  | The PA1 key.           |         |       |          |    |           |
| 6  | <i>mmmmmm)ToString</i> |         |       |          |    |           |
| 7  |                        |         |       |          |    |           |
| 8  | [C#]                   | public  | const | Keys     |    | PageDown; |
| 9  | [C++]                  | public: | const | Keys     |    | PageDown; |
| 10 | [VB]                   | Public  | Const | PageDown | As | Keys      |
| 11 | [JScript]              | public  | var   | PageDown | :  | Keys;     |
| 12 |                        |         |       |          |    |           |
| 13 | <i>Description</i>     |         |       |          |    |           |
| 14 | The PAGE DOWN key.     |         |       |          |    |           |
| 15 | <i>nnnnnn)ToString</i> |         |       |          |    |           |
| 16 |                        |         |       |          |    |           |
| 17 | [C#]                   | public  | const | Keys     |    | PageUp;   |
| 18 | [C++]                  | public: | const | Keys     |    | PageUp;   |
| 19 | [VB]                   | Public  | Const | PageUp   | As | Keys      |
| 20 | [JScript]              | public  | var   | PageUp   | :  | Keys;     |
| 21 |                        |         |       |          |    |           |
| 22 | <i>Description</i>     |         |       |          |    |           |
| 23 | The PAGE UP key.       |         |       |          |    |           |
| 24 |                        |         |       |          |    |           |
| 25 |                        |         |       |          |    |           |

oooooo)ToString

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Pause;  |
| [C++]     | public: | const | Keys  | Pause;  |
| [VB]      | Public  | Const | Pause | As Keys |
| [JScript] | public  | var   | Pause | : Keys; |

Description

The PAUSE key.

pppppp)ToString

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Play;   |
| [C++]     | public: | const | Keys | Play;   |
| [VB]      | Public  | Const | Play | As Keys |
| [JScript] | public  | var   | Play | : Keys; |

Description

The PLAY key.

qqqqqq)ToString

|           |         |       |       |         |
|-----------|---------|-------|-------|---------|
| [C#]      | public  | const | Keys  | Print;  |
| [C++]     | public: | const | Keys  | Print;  |
| [VB]      | Public  | Const | Print | As Keys |
| [JScript] | public  | var   | Print | : Keys; |

1  
2 *Description*

3 The PRINT key.

4 *rrrrrr) ToString*

5  
6 [C#] public const Keys PrintScreen;

7 [C++] public: const Keys PrintScreen;

8 [VB] Public Const PrintScreen As Keys

9 [JScript] public var PrintScreen : Keys;

10  
11 *Description*

12 The PRINT SCREEN key.

13 *ssssss) ToString*

14  
15 [C#] public const Keys Prior;

16 [C++] public: const Keys Prior;

17 [VB] Public Const Prior As Keys

18 [JScript] public var Prior : Keys;

19  
20 *Description*

21 The PAGE UP key.

22 *ttttt) ToString*

23  
24 [C#] public const Keys ProcessKey;

25 [C++] public: const Keys ProcessKey;



|   |           |        |       |            |    |       |
|---|-----------|--------|-------|------------|----|-------|
| 1 | [VB]      | Public | Const | ProcessKey | As | Keys  |
| 2 | [JScript] | public | var   | ProcessKey | :  | Keys; |

3

*Description*

The PROCESS KEY key.

*uuuuuu)ToString*

7

|   |      |        |       |      |    |
|---|------|--------|-------|------|----|
| 8 | [C#] | public | const | Keys | Q; |
|---|------|--------|-------|------|----|

|   |       |         |       |      |    |
|---|-------|---------|-------|------|----|
| 9 | [C++] | public: | const | Keys | Q; |
|---|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 10 | [VB] | Public | Const | Q | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 11 | [JScript] | public | var | Q | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

12

*Description*

The Q key.

*vvvvvv)ToString*

16

|    |      |        |       |      |    |
|----|------|--------|-------|------|----|
| 17 | [C#] | public | const | Keys | R; |
|----|------|--------|-------|------|----|

|    |       |         |       |      |    |
|----|-------|---------|-------|------|----|
| 18 | [C++] | public: | const | Keys | R; |
|----|-------|---------|-------|------|----|

|    |      |        |       |   |    |      |
|----|------|--------|-------|---|----|------|
| 19 | [VB] | Public | Const | R | As | Keys |
|----|------|--------|-------|---|----|------|

|    |           |        |     |   |   |       |
|----|-----------|--------|-----|---|---|-------|
| 20 | [JScript] | public | var | R | : | Keys; |
|----|-----------|--------|-----|---|---|-------|

21

*Description*

The R key.

24

25

*wwwww)ToString*

|           |         |       |         |          |
|-----------|---------|-------|---------|----------|
| [C#]      | public  | const | Keys    | RButton; |
| [C++]     | public: | const | Keys    | RButton; |
| [VB]      | Public  | Const | RButton | As Keys  |
| [JScript] | public  | var   | RButton | : Keys;  |

*Description*

The right mouse button.

*xxxxxx)ToString*

|           |         |       |             |              |
|-----------|---------|-------|-------------|--------------|
| [C#]      | public  | const | Keys        | RControlKey; |
| [C++]     | public: | const | Keys        | RControlKey; |
| [VB]      | Public  | Const | RControlKey | As Keys      |
| [JScript] | public  | var   | RControlKey | : Keys;      |

*Description*

The right CTRL key.

*yyyyyy)ToString*

|           |         |       |        |         |
|-----------|---------|-------|--------|---------|
| [C#]      | public  | const | Keys   | Return; |
| [C++]     | public: | const | Keys   | Return; |
| [VB]      | Public  | Const | Return | As Keys |
| [JScript] | public  | var   | Return | : Keys; |

*Description*

The RETURN key.

*zzzzzz) ToString*

[C#]            public            const            Keys            Right;

[C++]           public:            const            Keys            Right;

[VB]            Public            Const            Right           As            Keys

[JScript]       public            var            Right           :            Keys;

*Description*

The RIGHT ARROW key.

*aaaaaaa) ToString*

[C#]            public            const            Keys            RMenu;

[C++]           public:            const            Keys            RMenu;

[VB]            Public            Const            RMenu           As            Keys

[JScript]       public            var            RMenu           :            Keys;

*Description*

The right ALT key.

*bbbbbbb) ToString*

[C#]            public            const            Keys            RShiftKey;

[C++]           public:            const            Keys            RShiftKey;

|    |                                                          |         |       |           |    |       |
|----|----------------------------------------------------------|---------|-------|-----------|----|-------|
| 1  | [VB]                                                     | Public  | Const | RShiftKey | As | Keys  |
| 2  | [JScript]                                                | public  | var   | RShiftKey | :  | Keys; |
| 3  |                                                          |         |       |           |    |       |
| 4  | <i>Description</i>                                       |         |       |           |    |       |
| 5  | The right SHIFT key.                                     |         |       |           |    |       |
| 6  | <i>cccccc)ToString</i>                                   |         |       |           |    |       |
| 7  |                                                          |         |       |           |    |       |
| 8  | [C#]                                                     | public  | const | Keys      |    | RWin; |
| 9  | [C++]                                                    | public: | const | Keys      |    | RWin; |
| 10 | [VB]                                                     | Public  | Const | RWin      | As | Keys  |
| 11 | [JScript]                                                | public  | var   | RWin      | :  | Keys; |
| 12 |                                                          |         |       |           |    |       |
| 13 | <i>Description</i>                                       |         |       |           |    |       |
| 14 | The right Windows logo key (Microsoft Natural Keyboard). |         |       |           |    |       |
| 15 | <i>ddddddd)ToString</i>                                  |         |       |           |    |       |
| 16 |                                                          |         |       |           |    |       |
| 17 | [C#]                                                     | public  | const | Keys      |    | S;    |
| 18 | [C++]                                                    | public: | const | Keys      |    | S;    |
| 19 | [VB]                                                     | Public  | Const | S         | As | Keys  |
| 20 | [JScript]                                                | public  | var   | S         | :  | Keys; |
| 21 |                                                          |         |       |           |    |       |
| 22 | <i>Description</i>                                       |         |       |           |    |       |
| 23 | The S key.                                               |         |       |           |    |       |
| 24 |                                                          |         |       |           |    |       |
| 25 |                                                          |         |       |           |    |       |

1                    *eeeeeee)ToString*

2                    [C#]                    public                    const                    Keys                    Scroll;

3                    [C++]                    public:                    const                    Keys                    Scroll;

4                    [VB]                    Public                    Const                    Scroll                    As                    Keys

5                    [JScript]                    public                    var                    Scroll                    :                    Keys;

6

7                    *Description*

8                    The SCROLL LOCK key.

9

10                    *ffffff) ToString*

11                    [C#]                    public                    const                    Keys                    Select;

12                    [C++]                    public:                    const                    Keys                    Select;

13                    [VB]                    Public                    Const                    Select                    As                    Keys

14                    [JScript]                    public                    var                    Select                    :                    Keys;

15

16                    *Description*

17                    The SELECT key.

18

19                    *ggggggg)ToString*

20

21                    [C#]                    public                    const                    Keys                    SelectMedia;

22                    [C++]                    public:                    const                    Keys                    SelectMedia;

23                    [VB]                    Public                    Const                    SelectMedia                    As                    Keys

24                    [JScript]                    public                    var                    SelectMedia                    :                    Keys;

25

*Description*

The Select Media key (Windows 2000 or later).

*hhhhhhh)ToString*

|      |        |       |      |            |
|------|--------|-------|------|------------|
| [C#] | public | const | Keys | Separator; |
|------|--------|-------|------|------------|

|       |         |       |      |            |
|-------|---------|-------|------|------------|
| [C++] | public: | const | Keys | Separator; |
|-------|---------|-------|------|------------|

|      |        |       |           |    |      |
|------|--------|-------|-----------|----|------|
| [VB] | Public | Const | Separator | As | Keys |
|------|--------|-------|-----------|----|------|

|           |        |     |           |   |       |
|-----------|--------|-----|-----------|---|-------|
| [JScript] | public | var | Separator | : | Keys; |
|-----------|--------|-----|-----------|---|-------|

*Description*

The Separator key.

*iiiiiii) ToString*

|      |        |       |      |        |
|------|--------|-------|------|--------|
| [C#] | public | const | Keys | Shift; |
|------|--------|-------|------|--------|

|       |         |       |      |        |
|-------|---------|-------|------|--------|
| [C++] | public: | const | Keys | Shift; |
|-------|---------|-------|------|--------|

|      |        |       |       |    |      |
|------|--------|-------|-------|----|------|
| [VB] | Public | Const | Shift | As | Keys |
|------|--------|-------|-------|----|------|

|           |        |     |       |   |       |
|-----------|--------|-----|-------|---|-------|
| [JScript] | public | var | Shift | : | Keys; |
|-----------|--------|-----|-------|---|-------|

*Description*

The SHIFT modifier key.

*jjjjjjj) ToString*

|      |        |       |      |           |
|------|--------|-------|------|-----------|
| [C#] | public | const | Keys | ShiftKey; |
|------|--------|-------|------|-----------|

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C++] | public: | const | Keys | ShiftKey; |
|-------|---------|-------|------|-----------|

|    |                          |         |       |          |    |           |
|----|--------------------------|---------|-------|----------|----|-----------|
| 1  | [VB]                     | Public  | Const | ShiftKey | As | Keys      |
| 2  | [JScript]                | public  | var   | ShiftKey | :  | Keys;     |
| 3  |                          |         |       |          |    |           |
| 4  | <i>Description</i>       |         |       |          |    |           |
| 5  | The SHIFT key.           |         |       |          |    |           |
| 6  | <i>kkkkkkk)ToString</i>  |         |       |          |    |           |
| 7  |                          |         |       |          |    |           |
| 8  | [C#]                     | public  | const | Keys     |    | Snapshot; |
| 9  | [C++]                    | public: | const | Keys     |    | Snapshot; |
| 10 | [VB]                     | Public  | Const | Snapshot | As | Keys      |
| 11 | [JScript]                | public  | var   | Snapshot | :  | Keys;     |
| 12 |                          |         |       |          |    |           |
| 13 | <i>Description</i>       |         |       |          |    |           |
| 14 | The PRINT SCREEN key.    |         |       |          |    |           |
| 15 | <i>lllllll) ToString</i> |         |       |          |    |           |
| 16 |                          |         |       |          |    |           |
| 17 | [C#]                     | public  | const | Keys     |    | Space;    |
| 18 | [C++]                    | public: | const | Keys     |    | Space;    |
| 19 | [VB]                     | Public  | Const | Space    | As | Keys      |
| 20 | [JScript]                | public  | var   | Space    | :  | Keys;     |
| 21 |                          |         |       |          |    |           |
| 22 | <i>Description</i>       |         |       |          |    |           |
| 23 | The SPACEBAR key.        |         |       |          |    |           |
| 24 |                          |         |       |          |    |           |
| 25 |                          |         |       |          |    |           |

***mmmmmm)ToString***

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | Subtract; |
| [C++]     | public: | const | Keys     | Subtract; |
| [VB]      | Public  | Const | Subtract | As Keys   |
| [JScript] | public  | var   | Subtract | : Keys;   |

***Description***

The Subtract key.

***nnnnnnn)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | T;      |
| [C++]     | public: | const | Keys | T;      |
| [VB]      | Public  | Const | T    | As Keys |
| [JScript] | public  | var   | T    | : Keys; |

***Description***

The T key.

***ooooooo)ToString***

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | Tab;    |
| [C++]     | public: | const | Keys | Tab;    |
| [VB]      | Public  | Const | Tab  | As Keys |
| [JScript] | public  | var   | Tab  | : Keys; |



*Description*

The TAB key.

*ppppppp)ToString*

|      |        |       |      |    |
|------|--------|-------|------|----|
| [C#] | public | const | Keys | U; |
|------|--------|-------|------|----|

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C++] | public: | const | Keys | U; |
|-------|---------|-------|------|----|

|      |        |       |   |    |      |
|------|--------|-------|---|----|------|
| [VB] | Public | Const | U | As | Keys |
|------|--------|-------|---|----|------|

|           |        |     |   |   |       |
|-----------|--------|-----|---|---|-------|
| [JScript] | public | var | U | : | Keys; |
|-----------|--------|-----|---|---|-------|

*Description*

The U key.

*qqqqqqq)ToString*

|      |        |       |      |     |
|------|--------|-------|------|-----|
| [C#] | public | const | Keys | Up; |
|------|--------|-------|------|-----|

|       |         |       |      |     |
|-------|---------|-------|------|-----|
| [C++] | public: | const | Keys | Up; |
|-------|---------|-------|------|-----|

|      |        |       |    |    |      |
|------|--------|-------|----|----|------|
| [VB] | Public | Const | Up | As | Keys |
|------|--------|-------|----|----|------|

|           |        |     |    |   |       |
|-----------|--------|-----|----|---|-------|
| [JScript] | public | var | Up | : | Keys; |
|-----------|--------|-----|----|---|-------|

*Description*

The UP ARROW key.

*rrrrrrr)ToString*

|      |        |       |      |    |
|------|--------|-------|------|----|
| [C#] | public | const | Keys | V; |
|------|--------|-------|------|----|

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C++] | public: | const | Keys | V; |
|-------|---------|-------|------|----|

|           |        |       |   |    |       |
|-----------|--------|-------|---|----|-------|
| [VB]      | Public | Const | V | As | Keys  |
| [JScript] | public | var   | V | :  | Keys; |

#### Description

The V key.

*sssssss)ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | VolumeDown; |
| [C++]     | public: | const | Keys       | VolumeDown; |
| [VB]      | Public  | Const | VolumeDown | As Keys     |
| [JScript] | public  | var   | VolumeDown | : Keys;     |

#### Description

The Volume Down key (Windows 2000 or later).

*tttttt) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | Keys       | VolumeMute; |
| [C++]     | public: | const | Keys       | VolumeMute; |
| [VB]      | Public  | Const | VolumeMute | As Keys     |
| [JScript] | public  | var   | VolumeMute | : Keys;     |

#### Description

The Volume Mute key (Windows 2000 or later).

*uuuuuuu)ToString*

|           |         |       |          |           |
|-----------|---------|-------|----------|-----------|
| [C#]      | public  | const | Keys     | VolumeUp; |
| [C++]     | public: | const | Keys     | VolumeUp; |
| [VB]      | Public  | Const | VolumeUp | As Keys   |
| [JScript] | public  | var   | VolumeUp | : Keys;   |

*Description*

The Volume Up key (Windows 2000 or later).

*vvvvvvv)ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | W;      |
| [C++]     | public: | const | Keys | W;      |
| [VB]      | Public  | Const | W    | As Keys |
| [JScript] | public  | var   | W    | : Keys; |

*Description*

The W key.

*wwwwwww)ToString*

|           |         |       |      |         |
|-----------|---------|-------|------|---------|
| [C#]      | public  | const | Keys | X;      |
| [C++]     | public: | const | Keys | X;      |
| [VB]      | Public  | Const | X    | As Keys |
| [JScript] | public  | var   | X    | : Keys; |

*Description*

The X key.

*xxxxxxx)ToString*

|      |        |       |      |           |
|------|--------|-------|------|-----------|
| [C#] | public | const | Keys | XButton1; |
|------|--------|-------|------|-----------|

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C++] | public: | const | Keys | XButton1; |
|-------|---------|-------|------|-----------|

|      |        |       |          |    |      |
|------|--------|-------|----------|----|------|
| [VB] | Public | Const | XButton1 | As | Keys |
|------|--------|-------|----------|----|------|

|           |        |     |          |   |       |
|-----------|--------|-----|----------|---|-------|
| [JScript] | public | var | XButton1 | : | Keys; |
|-----------|--------|-----|----------|---|-------|

*Description*

The first x mouse button (five-button mouse).

*yyyyyyy)ToString*

|      |        |       |      |           |
|------|--------|-------|------|-----------|
| [C#] | public | const | Keys | XButton2; |
|------|--------|-------|------|-----------|

|       |         |       |      |           |
|-------|---------|-------|------|-----------|
| [C++] | public: | const | Keys | XButton2; |
|-------|---------|-------|------|-----------|

|      |        |       |          |    |      |
|------|--------|-------|----------|----|------|
| [VB] | Public | Const | XButton2 | As | Keys |
|------|--------|-------|----------|----|------|

|           |        |     |          |   |       |
|-----------|--------|-----|----------|---|-------|
| [JScript] | public | var | XButton2 | : | Keys; |
|-----------|--------|-----|----------|---|-------|

*Description*

The second x mouse button (five-button mouse).

*zzzzzzz)ToString*

|      |        |       |      |    |
|------|--------|-------|------|----|
| [C#] | public | const | Keys | Y; |
|------|--------|-------|------|----|

|       |         |       |      |    |
|-------|---------|-------|------|----|
| [C++] | public: | const | Keys | Y; |
|-------|---------|-------|------|----|

|   |           |        |       |   |    |       |
|---|-----------|--------|-------|---|----|-------|
| 1 | [VB]      | Public | Const | Y | As | Keys  |
| 2 | [JScript] | public | var   | Y | :  | Keys; |

3

4 *Description*

5 The Y key.

6 *aaaaaaaa)ToString*

|    |           |         |       |   |      |       |
|----|-----------|---------|-------|---|------|-------|
| 8  | [C#]      | public  | const |   | Keys | Z;    |
| 9  | [C++]     | public: | const |   | Keys | Z;    |
| 10 | [VB]      | Public  | Const | Z | As   | Keys  |
| 11 | [JScript] | public  | var   | Z | :    | Keys; |

12

13 *Description*

14 The Z key.

15 *bbbbbbbb)ToString*

|    |           |         |       |      |      |       |
|----|-----------|---------|-------|------|------|-------|
| 17 | [C#]      | public  | const |      | Keys | Zoom; |
| 18 | [C++]     | public: | const |      | Keys | Zoom; |
| 19 | [VB]      | Public  | Const | Zoom | As   | Keys  |
| 20 | [JScript] | public  | var   | Zoom | :    | Keys; |

21

22 *Description*

23 The ZOOM key.

24

25

KeysConverter class (System.Windows.Forms)

a) *ToString*

*Description*

Provides a **System.ComponentModel.TypeConverter** to convert **System.Windows.Forms.Keys** objects to and from other representations.

b) *KeysConverter*

*Example Syntax:*

c) *ToString*

[C#] public KeysConverter();

[C++] public: KeysConverter();

[VB] Public Sub New()

[JScript] public function KeysConverter();

d) *CanConvertFrom*

[C#] public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType);

[C++] public: bool CanConvertFrom(ITypeDescriptorContext\* context, Type\* sourceType);

[VB] Overrides Public Function CanConvertFrom(ByVal context As ITypeDescriptorContext, ByVal sourceType As Type) As Boolean

[JScript] public override function CanConvertFrom(context : ITypeDescriptorContext, sourceType : Type) : Boolean; Geta a value indicating

whether this converter can convert an object in the Specified source type to the native type of the converter.

#### *Description*

Gets a value indicating whether this converter can convert an object in the specified source type to the native type of the converter using the specified context.

*Return Value:* **true** if this object can perform the conversion; otherwise, **false** .

Override this method to provide your own conversion requirements. An *ITypeDescriptorContext* that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null . The type to convert from.

#### *e) Compare*

[C#] public int Compare(object a, object b);

[C++] public: \_\_sealed int Compare(Object\* a, Object\* b);

[VB] NotOverridable Public Function Compare(ByVal a As Object, ByVal b As Object) As Integer

[JScript] public function Compare(a : Object, b : Object) : int;

#### *Description*

Compares two key values for equivalence.

*Return Value:* An integer indicating the relationship between the two comparands.

This method uses **System.String.Compare(System.String, System.String)** to compare the two objects. An **System.Object** that represents the first key to compare. An **System.Object** that represents the second key to compare.

*f) ConvertFrom*

```
[C#] public override object ConvertFrom(ITypeDescriptorContext context,
CultureInfo culture, object value);
[C++] public: Object* ConvertFrom(ITypeDescriptorContext* context,
CultureInfo* culture, Object* value);
[VB] Overrides Public Function ConvertFrom(ByVal context As
ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object)
As Object
[JavaScript] public override function ConvertFrom(context : ITypeDescriptorContext,
culture : CultureInfo, value : Object) : Object; Converts the specified object to the
native type of the converter.
```

*Description*

Converts the specified object to the converter's native type.

*Return Value:* An Object that represents the converted *value* .

The context parameter can be used to extract additional information about the environment this converter is being invoked from. This may be **null** , so you should always check. Also, properties on the context object may also return null . An ITypeDescriptorContext that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null . A CultureInfo object to provide locale information. The object to convert.

*g) ConvertTo*

```
[C#] public override object ConvertTo(ITypeDescriptorContext context,
CultureInfo culture, object value, Type destinationType);
[C++] public: Object* ConvertTo(ITypeDescriptorContext* context, CultureInfo*
```



```

1 culture, Object* value, Type* destinationType);
2 [VB] Overrides Public Function ConvertTo(ByVal context As
3 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,
4 ByVal destinationType As Type) As Object
5 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,
6 culture : CultureInfo, value : Object, destinationType : Type) : Object; Converts
7 the Specified object object to the specified destination type.
8

```

#### 9 *Description*

10 Converts the specified object to the specified destination type.

*Return Value:* An Object that represents the converted *value* .

11 Override this method to provide your own conversion requirements. An  
12 ITypeDescriptorContext that provides a format context, which can be used to  
13 extract additional information about the environment this converter is being  
14 invoked from. This parameter or properties of this parameter can be null . A  
15 CultureInfo object to provide locale information. The object to convert. The type  
16 to convert the object to.

#### 15 *h) GetStandardValues*

```

17 [C#] public override StandardValuesCollection
18 GetStandardValues(ITypeDescriptorContext context);
19 [C++] public: StandardValuesCollection*
20 GetStandardValues(ITypeDescriptorContext* context);
21 [VB] Overrides Public Function GetStandardValues(ByVal context As
22 ITypeDescriptorContext) As StandardValuesCollection
23 [JScript] public override function GetStandardValues(context :
24 ITypeDescriptorContext) : StandardValuesCollection; Returns a collection of
25

```

standard values for the data type that this type converter is designed for.

### *Description*

Returns a collection of standard values for the data type that this type converter is designed for when provided with a format context.

*Return Value:* A StandardValuesCollection that holds a standard set of valid values, or null if the data type does not support a standard set of values.

The collection returned contains the values of the keys that can be converted. An ITypeDescriptorContext that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null .

#### *i) GetStandardValuesExclusive*

```
[C#] public override bool GetStandardValuesExclusive(ITypeDescriptorContext  
context);
```

```
[C++] public: bool GetStandardValuesExclusive(ITypeDescriptorContext*  
context);
```

```
[VB] Overrides Public Function GetStandardValuesExclusive(ByVal context As  
ITypeDescriptorContext) As Boolean
```

```
[JScript] public override function GetStandardValuesExclusive(context :  
ITypeDescriptorContext) : Boolean;
```

### *Description*

Determines if the list of standard values returned from GetStandardValues is an exclusive list. If the list is exclusive, then no other values are valid, such as in an enum data type. If the list is not exclusive, then there are other valid values besides the list of standard values GetStandardValues provides.

*Return Value:* True if the collection returned from GetStandardValues is an exhaustive list of possible values, or false if other values are possible. The default for this method always returns false; Determines if the list of standard values returned from GetStandardValues is an exclusive list. If the list is

exclusive, then no other values are valid, such as in an enum data type. If the list is not exclusive, then there are other valid values besides the list of standard values `GetStandardValues` provides. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null.

j) *GetStandardValuesSupported*

[C#] public override bool GetStandardValuesSupported(ITypeDescriptorContext context);

[C++] public: bool GetStandardValuesSupported(ITypeDescriptorContext\* context);

[VB] Overrides Public Function GetStandardValuesSupported(ByVal context As ITypeDescriptorContext) As Boolean

[JScript] public override function GetStandardValuesSupported(context : ITypeDescriptorContext) : Boolean; Gets a value indicating whether this object supports a standard set of values that can be picked from a list.

*Description*

Gets a value indicating whether this object supports a standard set of values that can be picked from a list.

*Return Value:* Always returns **true** . An `ITypeDescriptorContext` that provides a format context, which can be used to extract additional information about the environment this converter is being invoked from. This parameter or properties of this parameter can be null.

Label class (System.Windows.Forms)

*a) ToString*

*Description*

Represents a standard Windows label.

**System.Windows.Forms.Label** controls are typically used to provide descriptive text for a control. For example, you can use a **System.Windows.Forms.Label** to add descriptive text for a **System.Windows.Forms.TextBox** control to inform the user about the type of data expected in the control.

*b) Label*

*Example Syntax:*

*c) ToString*

|           |         |                   |
|-----------|---------|-------------------|
| [C#]      | public  | Label();          |
| [C++]     | public: | Label();          |
| [VB]      | Public  | Sub New()         |
| [JScript] | public  | function Label(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.Label** class.

By default, a label is displayed with its **System.Windows.Forms.Label.AutoSize** property set to **false** and with its **System.Windows.Forms.Label.BorderStyle** property set to **BorderStyle.None** .

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AutoSize*
- l) *ToString*

#### *Description*

Gets or sets a value indicating whether the control is automatically resized to display its entire contents.

When this property is set to **true**, the control's height and width are automatically adjusted to display the entire contents of the control. This property is typically set to **true** when you use a **System.Windows.Forms.Label** control to display various lengths of text, such as the status of an application process. You can also use this property when the application will display text in various languages, and the size of the text may increase or decrease based on the language settings in Windows.

- m) *BackColor*
- n) *BackgroundImage*
- o) *ToString*

#### *Description*

Gets or sets the image rendered on the background of the control.

*p)      **BindingContext***

*q)      **BorderStyle***

*r)      **ToString***

#### *Description*

Gets or sets the border style for the control.

You can use this property to add a border to the control. This property is typically used to differentiate a **System.Windows.Forms.Label** that labels another control from a **System.Windows.Forms.Label** that displays the status of a process in an application.

- s) *Bottom*
- t) *Bounds*
- u) *CanFocus*
- v) *CanSelect*
- w) *Capture*
- x) *CausesValidation*
- y) *ClientRectangle*
- z) *ClientSize*
- aa) *CompanyName*
- bb) *Container*
- cc) *ContainsFocus*
- dd) *ContextMenu*
- ee) *Controls*
- ff) *Created*
- gg) *CreateParams*
- hh) *ToString*

## *Description*

Overrides Control. A Label is a Win32 STATIC control, which we setup here.

the item that you are going to draw in your event handler for the **System.Windows.Forms.ListBox.DrawItem** event.

*zzzzz) ToString*

#### *Description*

Occurs when an owner-drawn **System.Windows.Forms.ListBox** is created and the sizes of the list items are determined.

You can create an event handler for this event to specify the size an item will be made before it is drawn in the **System.Windows.Forms.ListBox.DrawItem** event. The event is only raised when the **System.Windows.Forms.ListBox.DrawMode** property is set **DrawMode.OwnerDrawVariable**.

*aaaaaa) ToString*

#### *Description*

Occurs when the **System.Windows.Forms.ListBox.SelectedIndex** property has changed.

You can create an event handler for this event to determine when the selected index in the **System.Windows.Forms.ListBox** has been changed. This can be useful when you need to display information in other controls based on the current selection in the **System.Windows.Forms.ListBox**. You can use the event handler for this event to load the information in the other controls.

*bbbbbb) AddItemsCore*

[C#]      protected      virtual      void      AddItemsCore(object[]      value);

[C++]    protected:    virtual    void    AddItemsCore(Object\*    value    \_\_gc[]);

[VB]    Overridable    Protected    Sub    AddItemsCore(ByVal value() As Object)

[JScript]      protected      function      AddItemsCore(value      :      Object[]);



## Description

Adds an array of items to the **System.Windows.Forms.ListBox** .

You can use this method in your derived classes to add items to the item collection. An array of objects to add to the control.

## *cccccc)BeginUpdate*

|           |         |          |                |
|-----------|---------|----------|----------------|
| [C#]      | public  | void     | BeginUpdate(); |
| [C++]     | public: | void     | BeginUpdate(); |
| [VB]      | Public  | Sub      | BeginUpdate()  |
| [JScript] | public  | function | BeginUpdate(); |

## Description

Maintains performance while items are added to the **System.Windows.Forms.ListBox** one at a time by preventing the control from drawing until the **System.Windows.Forms.ListBox.EndUpdate** method is called.

The preferred way to add multiple items to the **System.Windows.Forms.ListBox** is to use the **System.Windows.Forms.ListBox.ObjectCollection.AddRange(System.Windows.Forms.ListBox.ObjectCollection)** method of the **System.Windows.Forms.ListBox.ObjectCollection** class (through the **System.Windows.Forms.ListBox.Items** property of the **System.Windows.Forms.ListBox** ). This enables you to add an array of items to the list in a single operation. However, if you want to add items one at a time using the **System.Windows.Forms.ListBox.ObjectCollection.Add(System.Object)** method of the **System.Windows.Forms.ListBox.ObjectCollection** class, you can use the **System.Windows.Forms.ListBox.BeginUpdate** method to prevent the control from repainting the **System.Windows.Forms.ListBox** each time an item is added to the list. Once you have completed the task of adding items to the list, call the **System.Windows.Forms.ListBox.EndUpdate** method to enable the **System.Windows.Forms.ListBox** to repaint. This way of adding items can

prevent flickered drawing of the **System.Windows.Forms.ListBox** when a large number of items are being added to the list.

#### *dddddd)ClearSelected*

|           |         |          |                  |
|-----------|---------|----------|------------------|
| [C#]      | public  | void     | ClearSelected(); |
| [C++]     | public: | void     | ClearSelected(); |
| [VB]      | Public  | Sub      | ClearSelected()  |
| [JScript] | public  | function | ClearSelected(); |

#### *Description*

Unselects all items in the **System.Windows.Forms.ListBox** .

Calling this method is equivalent to setting the **System.Windows.Forms.ListBox.SelectedIndex** property to negative one (-1). You can use this method to quickly remove the selection from all items in the list.

#### *eeeeee)CreateItemCollection*

|           |                                                                           |          |                        |                         |
|-----------|---------------------------------------------------------------------------|----------|------------------------|-------------------------|
| [C#]      | protected                                                                 | virtual  | ObjectCollection       | CreateItemCollection(); |
| [C++]     | protected:                                                                | virtual  | ObjectCollection*      | CreateItemCollection(); |
| [VB]      | Overridable Protected Function CreateItemCollection() As ObjectCollection |          |                        |                         |
| [JScript] | protected                                                                 | function | CreateItemCollection() | : ObjectCollection;     |

#### *Description*

Creates a new instance of the item collection.

*Return Value:* A **System.Windows.Forms.ListBox.ObjectCollection** that represents the new item collection.

You can override this in your derived classes to provide a different collection to store your items.

### *fffff) EndUpdate*

|           |         |          |              |
|-----------|---------|----------|--------------|
| [C#]      | public  | void     | EndUpdate(); |
| [C++]     | public: | void     | EndUpdate(); |
| [VB]      | Public  | Sub      | EndUpdate()  |
| [JScript] | public  | function | EndUpdate(); |

### *Description*

Resumes painting the **System.Windows.Forms.ListBox** control after painting is suspended by the **System.Windows.Forms.ListBox.BeginUpdate** method.

The preferred way to add items to the **System.Windows.Forms.ListBox** is to use the **System.Windows.Forms.ListBox.ObjectCollection.AddRange(System.Windows.Forms.ListBox.ObjectCollection)** method of the **System.Windows.Forms.ListBox.ObjectCollection** class (through the **System.Windows.Forms.ListBox.Items** property of the **System.Windows.Forms.ListBox** ). This enables you to add an array of items to the list at one time. However, if you want to add items one at a time using the **System.Windows.Forms.ListBox.ObjectCollection.Add(System.Object)** method of the **System.Windows.Forms.ListBox.ObjectCollection** class, you can use the **System.Windows.Forms.ListBox.BeginUpdate** method to prevent the control from repainting the **System.Windows.Forms.ListBox** each time an item is added to the list. Once you have completed the task of adding items to the list, call the **System.Windows.Forms.ListBox.EndUpdate** method to enable the **System.Windows.Forms.ListBox** to repaint. This way of adding items can prevent flickered drawing of the **System.Windows.Forms.ListBox** when a large number of items are being added to the list.

### *ggggg) FindString*

|       |         |          |                               |            |
|-------|---------|----------|-------------------------------|------------|
| [C#]  | public  | int      | FindString(string             | s);        |
| [C++] | public: | int      | FindString(String*            | s);        |
| [VB]  | Public  | Function | FindString(ByVal s As String) | As Integer |

[JScript] public function FindString(s : String) : int; Finds the first item in the **System.Windows.Forms.ListBox** that starts with the specified string.

#### *Description*

Finds the first item in the **System.Windows.Forms.ListBox** that starts with the specified string.

*Return Value:* The zero-based index of the first item found; returns **ListBox.NoMatches** if no match is found.

The search performed by this method is not case-sensitive. The search looks for words that partially match the specified search string parameter, *s*. You can use this method to search for the first item that matches the specified string. You can then perform tasks such as removing the item that contains the search text by using the

**System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. Once you have found the specified text, if you want to search for other instances of the text in the **System.Windows.Forms.ListBox**, you can use the version of the **System.Windows.Forms.ListBox.FindString(System.String)** method that provides a parameter for specifying a starting index within the **System.Windows.Forms.ListBox**. If you want to perform a search for an exact word match instead of a partial match, use the **System.Windows.Forms.ListBox.FindStringExact(System.String)** method. The text to search for.

#### *hhhhh)FindString*

[C#] public int FindString(string s, int startIndex);

[C++] public: int FindString(String\* s, int startIndex);

[VB] Public Function FindString(ByVal s As String, ByVal startIndex As Integer)

As Integer

[JScript] public function FindString(s : String, startIndex : int) : int;

#### *Description*

Finds the first item in the **System.Windows.Forms.ListBox** that starts with the specified string. The search starts at a specific starting index.

*Return Value:* The zero-based index of the first item found; returns **ListBox.NoMatches** if no match is found.

The search performed by this method is not case-sensitive. The search looks for words that partially match the specified search string parameter, *s*. You can use this method to search for the first item that matches the specified string at the specified starting index within the list of items for the

**System.Windows.Forms.ListBox**. You can then perform tasks such as removing the item that contains the search text by using the **System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. This method is typically used after a call has been made using the version of this method that does not specify a starting index. Once an initial item has been found in the list, this method is typically used to find further instances of the search text by specifying the index position in the *startIndex* parameter of the item after the first found instance of the search text. If you want to perform a search for an exact word match instead of a partial match, use the

**System.Windows.Forms.ListBox.FindStringExact(System.String)** method. The text to search for. The zero-based index of the item before the first item to be searched. Set to negative one (-1) to search from the beginning of the control.

#### *iiiiii) FindStringExact*

[C#]            public            int            FindStringExact(string            s);

[C++]           public:            int            FindStringExact(String\*            s);

[VB]   Public   Function   FindStringExact(ByVal s As String) As Integer

[JScript] public function FindStringExact(s : String) : int; Finds the first item in the **System.Windows.Forms.ListBox** that exactly matches the specified string.

#### *Description*

Finds the first item in the **System.Windows.Forms.ListBox** that exactly matches the specified string.

*Return Value:* The zero-based index of the first item found; returns **ListBox.NoMatches** if no match is found.

The search performed by this method is not case-sensitive. The search looks for an exact match to the words specified in the search string parameter, *s*. You can use this method to search for the first item that matches the specified string. You can then perform tasks such as removing the item that contains the search text by using the

**System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object)** method or changing the item's text. Once you have found the specified text, if you want to search for other instances of the text in the **System.Windows.Forms.ListBox**, you can use the version of the **System.Windows.Forms.ListBox.FindStringExact(System.String)** method that provides a parameter for specifying a starting index within the **System.Windows.Forms.ListBox**. If you want to perform partial word search instead of an exact word match, use the **System.Windows.Forms.ListBox.FindString(System.String)** method. The text to search for.

### jjjjj) FindStringExact

```
[C#] public int FindStringExact(string s, int startIndex);
```

```
[C++] public: int FindStringExact(String* s, int startIndex);
```

```
[VB] Public Function FindStringExact(ByVal s As String, ByVal startIndex As Integer) As Integer
```

```
[JScript] public function FindStringExact(s : String, startIndex : int) : int;
```

*Description*

Finds the first item in the **System.Windows.Forms.ListBox** that exactly matches the specified string. The search starts at a specific starting index.

**Return Value:** The zero-based index of the first item found; returns **ListBox.NoMatches** if no match is found.

The search performed by this method is not case-sensitive. The search looks for words that exactly match the specified search string parameter, *s*. You can use this method to search for the first item that matches the specified string at the specified starting index within the list of items for the **System.Windows.Forms.ListBox**. You can then perform tasks such as removing the item that contains the search text using the **System.Windows.Forms.ListBox.ObjectCollection.Remove(System.Object)** method or change the item's text. This method is typically used after a call

has been made using the version of this method that does not specify a starting index. Once an initial item has been found in the list, this method is typically used to find further instances of the search text by specifying the index position in the *startIndex* parameter of the item after the first found instance of the search text. If you want to perform a partial word search instead of an exact word match, use the **System.Windows.Forms.ListBox.FindString(System.String)** method. The text to search for. The zero-based index of the item before the first item to be searched. Set to negative one (-1) to search from the beginning of the control.

#### *kkkkkk)GetItemHeight*

```
[C#]          public          int          GetItemHeight(int          index);
[C++]          public:          int          GetItemHeight(int          index);
[VB] Public Function GetItemHeight(ByVal index As Integer) As Integer
[JScript]      public      function      GetItemHeight(index      :      int)      :      int;
```

#### *Description*

Returns the height of an item in the **System.Windows.Forms.ListBox** .  
*Return Value:* The height, in pixels, of the specified item.

If the **System.Windows.Forms.ListBox.DrawMode** property is not set to **DrawMode.OwnerDrawVariable** , the value of the index parameter is ignored because all items in a standard **System.Windows.Forms.ListBox** are the same size. You can use this property when you are using an owner-drawn **System.Windows.Forms.ListBox** to determine the size of any item within the **System.Windows.Forms.ListBox** . The zero-based index of the item to return the height for.

#### *lllll) GetItemRectangle*

```
[C#]          public          Rectangle      GetItemRectangle(int          index);
[C++]          public:          Rectangle      GetItemRectangle(int          index);
[VB] Public Function GetItemRectangle(ByVal index As Integer) As Rectangle
```

1 [JScript] public function GetItemRectangle(index : int) : Rectangle;

3 *Description*

4 Returns the bounding rectangle for an item in the  
5 **System.Windows.Forms.ListBox** .

6 *Return Value:* A **System.Drawing.Rectangle** that represents the bounding  
7 rectangle for the specified item.

8 If the item specified in the *index* parameter is not visible, the rectangle returned  
9 by this method will be outside the visible portion of the control. You can use this  
10 method to determine the size and position of an item within the list. To get the  
11 height of an item, especially a variable-height owner drawn list item, you can use  
12 the **System.Windows.Forms.ListBox.GetItemHeight(System.Int32)**  
13 method. The zero-based index of item whose bounding rectangle you want to  
14 return.

15 *mmmmmm)GetSelected*

16 [C#] public bool GetSelected(int index);

17 [C++] public: bool GetSelected(int index);

18 [VB] Public Function GetSelected(ByVal index As Integer) As Boolean

19 [JScript] public function GetSelected(index : int) : Boolean;

20 *Description*

21 Returns a value indicating whether the specified item is selected.

22 *Return Value:* **true** if the specified item is currently selected in the  
23 **System.Windows.Forms.ListBox** ; otherwise, **false** .

24 You can used this method to quickly determine whether a specified item is  
25 selected. This method is useful when a specific operation needs to be performed  
when a specific item in a multiple-selection **System.Windows.Forms.ListBox**  
is selected. The zero-based index of the item that determines whether it is  
selected.



## *nnnnnn)IndexFromPoint*

```
1
2 [C#]      public      int      IndexFromPoint(Point      p);
3
4 [C++]      public:      int      IndexFromPoint(Point      p);
5
6 [VB] Public Function IndexFromPoint(ByVal p As Point) As Integer
7
8 [JScript] public function IndexFromPoint(p : Point) : int; Returns the zero-based
9
10 index      of      the      item      at      the      specified      coordinates.
11
```

### *Description*

Returns the zero-based index of the item at the specified coordinates.

*Return Value:* The zero-based index of the item found at the specified coordinates; returns **ListBox.NoMatches** if no match is found.

This method enables you to determine which item is located at a specific location within the control. You can use this method to determine which item within the list is selected when a user right-clicks over the **System.Windows.Forms.ListBox**. The location of the cursor can be determined and passed to the *p* parameter of the **System.Windows.Forms.ListBox.IndexFromPoint(System.Drawing.Point)** method to determine which item the user right-clicked the mouse over. You can then display a shortcut menu to the user to provide tasks and features based on the specific item. A **System.Drawing.Point** object containing the coordinates used to obtain the item index.

## *oooooo)IndexFromPoint*

```
18
19
20 [C#]      public      int      IndexFromPoint(int      x,      int      y);
21
22 [C++]      public:      int      IndexFromPoint(int      x,      int      y);
23
24 [VB] Public Function IndexFromPoint(ByVal x As Integer, ByVal y As Integer)
25
26 As Integer
27
28 [JScript] public function IndexFromPoint(x : int, y : int) : int;
```

## Description

Returns the zero-based index of the item at the specified coordinates.

*Return Value:* The zero-based index of the item found at the specified coordinates; returns **ListBox.NoMatches** if no match is found.

This method enables you to determine which item that is located at a specific location within the control. You can use this method to determine which item within the list is selected when a user right-clicks over the **System.Windows.Forms.ListBox**. The location of the cursor can be determined and passed to the *x* and *y* parameters of the **System.Windows.Forms.ListBox.IndexFromPoint(System.Drawing.Point)** method to determine which item the user right-clicked the mouse over. You can then display a shortcut menu to the user to provide tasks and features based on the specific item. The *x* coordinate of the location to search. The *y* coordinate of the location to search.

## *pppppp)OnChangeUICues*

[C#] protected override void OnChangeUICues(UICuesEventArgs e);

[C++] protected: void OnChangeUICues(UICuesEventArgs\* e);

[VB] Overrides Protected Sub OnChangeUICues(ByVal e As UICuesEventArgs)

[JScript] protected override function OnChangeUICues(e : UICuesEventArgs);

## Description

## *qqqqqq)OnDataSourceChanged*

[C#] protected override void OnDataSourceChanged(EventArgs e);

[C++] protected: void OnDataSourceChanged(EventArgs\* e);

[VB] Overrides Protected Sub OnDataSourceChanged(ByVal e As EventArgs)

[JScript] protected override function OnDataSourceChanged(e : EventArgs);

*Description*

*rrrrrr) OnDisplayMemberChanged*

[C#] protected override void OnDisplayMemberChanged(EventArgs e);  
[C++] protected: void OnDisplayMemberChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnDisplayMemberChanged(ByVal e As  
EventArgs)  
[JScript] protected override function OnDisplayMemberChanged(e : EventArgs);

*Description*

*ssssss) OnDrawItem*

[C#] protected virtual void OnDrawItem(DrawItemEventArgs e);  
[C++] protected: virtual void OnDrawItem(DrawItemEventArgs\* e);  
[VB] Overridable Protected Sub OnDrawItem(ByVal e As DrawItemEventArgs)  
[JScript] protected function OnDrawItem(e : DrawItemEventArgs);

*Description*

Raises the **System.Windows.Forms.ListBox.DrawItem** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DrawItemEventArgs** that contains the event data.

### *ttttt) OnFontChanged*

```
[C#]    protected    override    void    OnFontChanged(EventArgs    e);
[C++]    protected:    void    OnFontChanged(EventArgs*    e);
[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)
[JScript] protected override function OnFontChanged(e : EventArgs);
```

### *Description*

### *uuuuuu)OnHandleCreated*

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

### *Description*

We need to know when the window handle has been created so we can set up a few things, like column width, etc! Inheriting classes should not forget to call `base.OnHandleCreated()`.

### *vvvvvv)OnHandleDestroyed*

```
[C#]    protected    override    void    OnHandleDestroyed(EventArgs    e);
[C++]    protected:    void    OnHandleDestroyed(EventArgs*    e);
[VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)
[JScript] protected override function OnHandleDestroyed(e : EventArgs);
```

## Description

Overridden to make sure that we set up and clear out items correctly. Inheriting controls should not forget to call `base.OnHandleDestroyed()` Overridden to make sure that we set up and clear out items correctly. Inheriting controls should not forget to call `base.OnHandleDestroyed()`

### *xxxxxxx)OnMeasureItem*

[C#] protected virtual void OnMeasureItem(MeasureItemEventArgs e);

[C++] protected: virtual void OnMeasureItem(MeasureItemEventArgs\* e);

[VB] Overridable Protected Sub OnMeasureItem(ByVal e As MeasureItemEventArgs)

[JScript] protected function OnMeasureItem(e : MeasureItemEventArgs);

## Description

Raises the **System.Windows.Forms.ListBox.MeasureItem** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MeasureItemEventArgs** that contains the event data.

### *xxxxxxx)OnParentChanged*

[C#] protected override void OnParentChanged(EventArgs e);

[C++] protected: void OnParentChanged(EventArgs\* e);

[VB] Overrides Protected Sub OnParentChanged(ByVal e As EventArgs)

[JScript] protected override function OnParentChanged(e : EventArgs);

## Description

We override this so we can re-create the handle if the parent has changed.

### *yyyyyy)OnResize*

```
[C#]      protected      override      void      OnResize(EventArgs      e);  
[C++]      protected:      void      OnResize(EventArgs*      e);  
[VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)  
[JScript] protected override function OnResize(e : EventArgs);
```

### *Description*

### *zzzzzz) OnSelectedIndexChanged*

```
[C#]      protected      override      void      OnSelectedIndexChanged(EventArgs      e);  
[C++]      protected:      void      OnSelectedIndexChanged(EventArgs*      e);  
[VB] Overrides Protected Sub OnSelectedIndexChanged(ByVal e As EventArgs)  
[JScript] protected override function OnSelectedIndexChanged(e : EventArgs);
```

### *Description*

Actually goes and fires the selectedIndexChanged event. Inheriting controls should use this to know when the event is fired [this is preferable to adding an event handler on yourself for this event]. They should, however, remember to call base.OnSelectedIndexChanged(e); to ensure the event is still fired to external listeners. Actually goes and fires the selectedIndexChanged event. Inheriting controls should use this to know when the event is fired [this is preferable to adding an event handler on yourself for this event]. They should, however, remember to call base.OnSelectedIndexChanged(e); to ensure the event is still fired to external listeners. Event object with the details

### ***aaaaaaa)RefreshItem***

```
1
2 [C#]      protected      override      void      RefreshItem(int      index);
3
4 [C++]      protected:      void      RefreshItem(int      index);
5
6 [VB] Overrides Protected Sub RefreshItem(ByVal index As Integer)
7
8 [JScript] protected override function RefreshItem(index : int);
9
```

#### ***Description***

Reparses the object at the given index, getting new text string for it.

### ***bbbbbbb)SetBoundsCore***

```
10
11 [C#] protected override void SetBoundsCore(int x, int y, int width, int height,
12 BoundsSpecified specified);
13
14 [C++] protected: void SetBoundsCore(int x, int y, int width, int height,
15 BoundsSpecified specified);
16
17 [VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As
18 Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As
19 BoundsSpecified)
20
21 [JScript] protected override function SetBoundsCore(x : int, y : int, width : int,
22 height : int, specified : BoundsSpecified);
23
24
25
```

#### ***Description***

Overrides Control.SetBoundsCore to remember the requestedHeight.

### *ccccccc)SetItemCore*

```
1
2 [C#] protected override void SetItemCore(int index, object value);
3 [C++] protected: void SetItemCore(int index, Object* value);
4 [VB] Overrides Protected Sub SetItemCore(ByVal index As Integer, ByVal value
5 As Object)
6 [JScript] protected override function SetItemCore(index : int, value : Object);
7
```

### *Description*

### *ddddddd)SetItemsCore*

```
12 [C#] protected override void SetItemsCore(object[] value);
13 [C++] protected: void SetItemsCore(Object* value __gc[]);
14 [VB] Overrides Protected Sub SetItemsCore(ByVal value() As Object)
15 [JScript] protected override function SetItemsCore(value : Object[]);
16
```

### *Description*

18 Clears the contents of the **System.Windows.Forms.ListBox** and adds the  
19 specified items to the control. An array of objects to insert into the control.

### *eeeeeee)SetSelected*

```
22 [C#] public void SetSelected(int index, bool value);
23 [C++] public: void SetSelected(int index, bool value);
24 [VB] Public Sub SetSelected(ByVal index As Integer, ByVal value As Boolean)
25
```



1 [JScript] public function SetSelected(index : int, value : Boolean);

3 *Description*

4 Selects or clears the selection for the specified item in a  
5 **System.Windows.Forms.ListBox** .

6 You can use this property to set the selection of items in a multiple-selection  
7 **System.Windows.Forms.ListBox** . To select an item in a single-selection  
8 **System.Windows.Forms.ListBox** , use the  
9 **System.Windows.Forms.ListBox.SelectedIndex** property. The zero-based  
10 index of the item in a **System.Windows.Forms.ListBox** to select or clear the  
11 selection for. **true** to select the specified item; otherwise, **false** .

9 *fffff) Sort*

|              |             |           |      |         |
|--------------|-------------|-----------|------|---------|
| 11 [C#]      | protected   | virtual   | void | Sort(); |
| 12 [C++]     | protected:  | virtual   | void | Sort(); |
| 13 [VB]      | Overridable | Protected | Sub  | Sort()  |
| 14 [JScript] | protected   | function  |      | Sort(); |

16 *Description*

17 Sorts the items in the **System.Windows.Forms.ListBox** alphabetically.

18 You can override this method in your derived class to provide your own sorting  
19 routine. When adding items to a **System.Windows.Forms.ListBox** , it is more  
20 efficient to sort the items first and then add new items.

20 *ggggggg) ToString*

|          |           |          |          |                      |
|----------|-----------|----------|----------|----------------------|
| 22 [C#]  | public    | override | string   | ToString();          |
| 23 [C++] | public:   |          | String*  | ToString();          |
| 24 [VB]  | Overrides | Public   | Function | ToString() As String |

1 [JScript] public override function ToString() : String;

3 *Description*

4 Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

5 **hhhhhhh)WmReflectCommand**

7 [C#] protected virtual void WmReflectCommand(ref Message m);

8 [C++] protected: virtual void WmReflectCommand(Message\* m);

9 [VB] Overridable Protected Sub WmReflectCommand(ByRef m As Message)

10 [JScript] protected function WmReflectCommand(m : Message);

12 *Description*

13 **iiiiiii) WndProc**

15 [C#] protected override void WndProc(ref Message m);

16 [C++] protected: void WndProc(Message\* m);

17 [VB] Overrides Protected Sub WndProc(ByRef m As Message)

18 [JScript] protected override function WndProc(m : Message);

20 *Description*

21 The list's window procedure. Inheriting classes can override this to add extra  
22 functionality, but should not forget to call base.wndProc(m); to ensure the list  
23 continues to function properly. A Windows Message Object.

ListControl class (System.Windows.Forms)

*a) WndProc*

*Description*

Provides a common implementation of members for the **System.Windows.Forms.ListBox** and **System.Windows.Forms.ComboBox** classes.

The **System.Windows.Forms.ListControl** class provides implementations of common elements for the **System.Windows.Forms.ListBox** and **System.Windows.Forms.ComboBox** controls.

*b) ListControl*

*Example Syntax:*

*c) WndProc*

|           |                    |                |
|-----------|--------------------|----------------|
| [C#]      | protected          | ListControl(); |
| [C++]     | protected:         | ListControl(); |
| [VB]      | Protected          | Sub New()      |
| [JScript] | protected function | ListControl(); |

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *BindingContext*
- n) *Bottom*
- o) *Bounds*
- p) *CanFocus*
- q) *CanSelect*
- r) *Capture*
- s) *CausesValidation*
- t) *ClientRectangle*
- u) *ClientSize*
- v) *CompanyName*
- w) *Container*
- x) *ContainsFocus*
- y) *ContextMenu*
- z) *Controls*

1        *aa) Created*

2        *bb) CreateParams*

3        *cc) Cursor*

4        *dd) DataBindings*

5        *ee) DataManager*

6        *ff) WndProc*

7  
8  
9        *Description*

10       Gets the **System.Windows.Forms.CurrencyManager** object associated with  
11       this control.

12       The **System.Windows.Forms.ListControl.DataManager** property is valid if  
13       the **System.Windows.Forms.ListControl.DataSource** property is set. If this  
14       is not a data-bound control, the default is **null** .

15        *gg) DataSource*

16        *hh) WndProc*

17       [C#]        public        object        DataSource        {get;        set;}

18       [C++]       public:    \_\_property   Object\*   get\_DataSource();public:   \_\_property   void  
19       set\_DataSource(Object\*);

20       [VB]        Public        Property        DataSource        As        Object

21       [JScript]   public   function   get   DataSource() :   Object;public   function   set  
22       DataSource(Object);

23  
24       *Description*

1 Gets or sets the data source for this **System.Windows.Forms.ListControl**  
object.

2 There are two ways to fill the **System.Windows.Forms.ComboBox** and  
3 **System.Windows.Forms.ListBox** controls.

4 *ii) DefaultImeMode*

5 *jj) DefaultSize*

6 *kk) DesignMode*

7 *ll) DisplayMember*

8 *mm) WndProc*

9  
10  
11 *Description*

12 Gets or sets a string that specifies the property of the data source whose  
13 contents to display.

14 The controls that inherit from **System.Windows.Forms.ListControl** can  
15 display diverse types of objects. If the specified property does not exist on the  
16 object or the value of **System.Windows.Forms.ListControl.DisplayMember**  
17 is an empty string (""), the results of the object's **ToString** method are  
18 displayed instead.  
19  
20  
21  
22  
23  
24  
25

1      **nn)    *DisplayRectangle***

2      **oo)    *Disposing***

3      **pp)    *Dock***

4      **qq)    *Enabled***

5      **rr)    *Events***

6      **ss)    *Focused***

7      **tt)    *Font***

8      **uu)    *FontHeight***

9      **vv)    *ForeColor***

10     **ww)    *Handle***

11     **xx)    *HasChildren***

12     **yy)    *Height***

13     **zz)    *ImeMode***

14     **aaa)    *InvokeRequired***

15     **bbb)    *IsAccessible***

16     **ccc)    *IsDisposed***

17     **ddd)    *IsHandleCreated***

18     **eee)    *Left***

19     **fff)    *Location***

20     **ggg)    *Name***

21     **hhh)    *Parent***

22     **iii)    *ProductName***

23     **jjj)    *ProductVersion***

*kkk) RecreatingHandle*

*lll) Region*

*mmm) RenderRightToLeft*

*nnn) ResizeRedraw*

*ooo) Right*

*ppp) RightToLeft*

*qqq) SelectedIndex*

*rrr) WndProc*

#### *Description*

When overridden in a derived class, gets or sets the zero-based index of the currently selected item.

*sss) SelectedValue*

*ttt) WndProc*

[C#]        public        object        SelectedValue        {get;        set;}

[C++] public: \_\_property Object\* get\_SelectedValue();public: \_\_property void  
set\_SelectedValue(Object\*);

[VB]        Public        Property        SelectedValue        As        Object

[JScript] public function get SelectedValue() : Object;public function set  
SelectedValue(Object);

#### *Description*



Gets or sets the value of the of the member property specified by the **System.Windows.Forms.ListControl.ValueMember** property.

If a property is not specified in **System.Windows.Forms.ListControl.ValueMember** , **System.Windows.Forms.ListControl.SelectedValue** returns the results of the **ToString** method of the object.

*uuu) ShowFocusCues*

*vvv) ShowKeyboardCues*

*www) Site*

*xxx) Size*

*yyy) TabIndex*

*zzz) TabStop*

*aaaa) Tag*

*bbbb) Text*

*cccc) Top*

*dddd) TopLevelControl*

*eeee) ValueMember*

*ffff) WndProc*

#### *Description*

Gets or sets a string that specifies the property of the data source from which to draw the value.

You specify the contents of the **System.Windows.Forms.ListControl.ValueMember** property in cases where you bind data.

1 *gggg) Visible*

2 *hhhh) Width*

3 *iiii) WindowTarget*

4 *jjjj) WndProc*

5  
6  
7 *Description*

8 Occurs when the **System.Windows.Forms.ListControl.DataSource** changes.

9 When you create a **System.Windows.Forms.ListControl** delegate, you  
10 identify the method that will handle the event. To associate the event with your  
11 event handler, add an instance of the delegate to the event. The event handler is  
called whenever the event occurs, unless you remove the delegate. For more  
information about event handler delegates, see .

12 *kkkk) WndProc*

13  
14 [C#] public event EventHandler DisplayMemberChanged;  
15 [C++] public: \_\_event EventHandler\* DisplayMemberChanged;  
16 [VB] Public Event DisplayMemberChanged As EventHandler

17  
18 *Description*

19 Occurs when the **System.Windows.Forms.ListControl.DisplayMember**  
property changes.

20 When you create a **System.Windows.Forms.ListControl** delegate, you  
21 identify the method that will handle the event. To associate the event with your  
22 event handler, add an instance of the delegate to the event. The event handler is  
23 called whenever the event occurs, unless you remove the delegate. For more  
24 information about event handler delegates, see .  
25

### III) *WndProc*

#### *Description*

Occurs when the **System.Windows.Forms.ListControl.SelectedValue** property changes.

When you create a **System.Windows.Forms.ListControl** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

#### *mmmm)WndProc*

#### *Description*

Occurs when the **System.Windows.Forms.ListControl.ValueMember** property changes.

When you create a **System.Windows.Forms.ListControl** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

#### *nnnn) FilterItemOnProperty*

[C#]       protected       object       FilterItemOnProperty(object       item);

[C++]       protected:       Object\*       FilterItemOnProperty(Object\*       item);

[VB] Protected Function FilterItemOnProperty(ByVal item As Object) As Object

[JScript] protected function FilterItemOnProperty(item : Object) : Object;

#### *Description*

oooo) *FilterItemOnProperty*

[C#] protected object FilterItemOnProperty(object item, string field);  
[C++] protected: Object\* FilterItemOnProperty(Object\* item, String\* field);  
[VB] Protected Function FilterItemOnProperty(ByVal item As Object, ByVal  
field As String) As Object  
[JScript] protected function FilterItemOnProperty(item : Object, field : String) :  
Object;

*Description*

pppp) *GetItemText*

[C#] public string GetItemText(object item);  
[C++] public: String\* GetItemText(Object\* item);  
[VB] Public Function GetItemText(ByVal item As Object) As String  
[JScript] public function GetItemText(item : Object) : String;

*Description*

Returns the text representation of the specified item.

*Return Value:* If the **System.Windows.Forms.ListControl.DisplayMember** property is **null**, the results of the object's **toString** method. Otherwise the results of the **toString** method of the member specified in the **System.Windows.Forms.ListControl.DisplayMember** property. The object from which to get the contents to display.

**qqqq) IsInputKey**

[C#] protected override bool IsInputKey(Keys keyData);  
[C++] protected: bool IsInputKey(Keys keyData);  
[VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As Boolean  
[JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

*Description*

Handling special input keys, such as pgup, pgdown, home, end, etc...

**rrrr) OnBindingContextChanged**

[C#] protected override void OnBindingContextChanged(EventArgs e);  
[C++] protected: void OnBindingContextChanged(EventArgs\* e);  
[VB] Overrides Protected Sub OnBindingContextChanged(ByVal e As EventArgs)  
[JScript] protected override function OnBindingContextChanged(e : EventArgs);

*Description*

**ssss) OnDataSourceChanged**

[C#] protected virtual void OnDataSourceChanged(EventArgs e);  
[C++] protected: virtual void OnDataSourceChanged(EventArgs\* e);  
[VB] Overridable Protected Sub OnDataSourceChanged(ByVal e As EventArgs)

1 [JScript] protected function OnDataSourceChanged(e : EventArgs);

2  
3 *Description*

4 Raises the **System.Windows.Forms.ListControl.DataSourceChanged** event.

5 Raising an event invokes the event handler through a delegate. For more  
6 information, see . An **System.EventArgs** that contains the event data.

7 *ttt) OnDisplayMemberChanged*

8  
9 [C#] protected virtual void OnDisplayMemberChanged(EventArgs e);

10 [C++] protected: virtual void OnDisplayMemberChanged(EventArgs\* e);

11 [VB] Overridable Protected Sub OnDisplayMemberChanged(ByVal e As  
12 EventArgs)

13 [JScript] protected function OnDisplayMemberChanged(e : EventArgs);

14  
15 *Description*

16 Raises the **System.Windows.Forms.ListControl.DisplayMemberChanged** event.

17 Raising an event invokes the event handler through a delegate. For more  
18 information, see . An **System.EventArgs** that contains the event data.

19 *uuuu) OnSelectedIndexChanged*

20  
21 [C#] protected virtual void OnSelectedIndexChanged(EventArgs e);

22 [C++] protected: virtual void OnSelectedIndexChanged(EventArgs\* e);

23 [VB] Overridable Protected Sub OnSelectedIndexChanged(ByVal e As  
24 EventArgs)

1 [JScript] protected function OnSelectedIndexChanged(e : EventArgs);

3 *Description*

4 Raises the **SelectedIndexChanged** event.

5 Raising an event invokes the event handler through a delegate. For more  
6 information, see . An **System.EventArgs** that contains the event data.

7 *vvvv) OnSelectedValueChanged*

8 [C#] protected virtual void OnSelectedValueChanged(EventArgs e);

9 [C++] protected: virtual void OnSelectedValueChanged(EventArgs\* e);

10 [VB] Overridable Protected Sub OnSelectedValueChanged(ByVal e As  
11 EventArgs)

12 [JScript] protected function OnSelectedValueChanged(e : EventArgs);

14 *Description*

15 Raises the **System.Windows.Forms.ListControl.SelectedValueChanged**  
16 event.

17 Raising an event invokes the event handler through a delegate. For more  
18 information, see . An **System.EventArgs** that contains the event data.

19 *www)OnValueMemberChanged*

20 [C#] protected virtual void OnValueMemberChanged(EventArgs e);

21 [C++] protected: virtual void OnValueMemberChanged(EventArgs\* e);

22 [VB] Overridable Protected Sub OnValueMemberChanged(ByVal e As  
23 EventArgs)

24 [JScript] protected function OnValueMemberChanged(e : EventArgs);

## Description

Raises the **System.Windows.Forms.ListControl.ValueMemberChanged** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

### xxxx) RefreshItem

```
[C#]      protected      abstract      void      RefreshItem(int      index);  
[C++]    protected:    virtual    void    RefreshItem(int    index)    =    0;  
[VB] MustOverride Protected Sub RefreshItem(ByVal index As Integer)  
[JScript] protected abstract function RefreshItem(index : int);
```

## Description

When overridden in a derived class, resynchronizes the data of the object at the specified index with the contents of the data source. The zero-based index of the item whose data to refresh.

### yyyy) SetItemCore

```
[C#]      protected      virtual      void      SetItemCore(int      index,      object      value);  
[C++]    protected:    virtual    void    SetItemCore(int    index,    Object*    value);  
[VB] Overridable Protected Sub SetItemCore(ByVal index As Integer, ByVal  
value                                     As                                     Object)  
[JScript] protected function SetItemCore(index : int, value : Object);
```

## Description



## zzzz) *SetItemsCore*

```
[C#]    protected    abstract    void    SetItemsCore(object[]    items);
[C++]   protected:   virtual    void    SetItemsCore(Object*    items    __gc[]) = 0;
[VB]    MustOverride Protected Sub SetItemsCore(ByVal items() As Object)
[JScript] protected    abstract    function    SetItemsCore(items    :    Object[]);
```

### *Description*

When overridden in a derived class, sets a array of objects in a collection in the derived class.

ListView class (System.Windows.Forms)

#### *a) WndProc*

### *Description*

Displays a list of items in one of four views. Each item displays a caption, and optionally, an image.

The **System.Windows.Forms.ListView.View** property specifies how to display the items: Large Icons Small Icons List Details Displays a list of items in one of four views. Each item displays a caption and optionally an image.

#### *b) ListView*

#### *Example Syntax:*

#### *c) WndProc*

```
[C#]                                public                                ListView();
[C++]                                public:                                ListView();
[VB]                                Public                                Sub                                New()
```

1 [JScript] public function ListView();

3 *Description*

4 Initializes a new instance of the **System.Windows.Forms.ListView** class with default styles.

- 5 d) *AccessibilityObject*
- 6 e) *AccessibleDefaultActionDescription*
- 7 f) *AccessibleDescription*
- 8 g) *AccessibleName*
- 9 h) *AccessibleRole*
- 10 i) *Activation*
- 11 j) *WndProc*

15 *Description*

16 Gets or sets the kind of user action required to activate an item.

- 17 k) *Alignment*
- 18 l) *WndProc*

20 [C#] public ListViewAlignment Alignment {get; set;}

21 [C++] public: \_\_property ListViewAlignment get\_Alignment();public: \_\_property

22 void set\_Alignment(ListViewAlignment);

23 [VB] Public Property Alignment As ListViewAlignment

24 [JScript] public function get Alignment() : ListViewAlignment;public function set

1 Alignment(ListViewAlignment);

2  
3 *Description*

4 Gets or sets the default window side that items are aligned to.

5 *m) AllowColumnReorder*

6 *n) WndProc*

7  
8 [C#] public bool AllowColumnReorder {get; set;}

9 [C++] public: \_\_property bool get\_AllowColumnReorder();public: \_\_property

10 void set\_AllowColumnReorder(bool);

11 [VB] Public Property AllowColumnReorder As Boolean

12 [JScript] public function get AllowColumnReorder() : Boolean;public function set

13 AllowColumnReorder(Boolean);

14  
15 *Description*

16 Gets or sets whether the user can drag column headers to other column  
17 positions, thus changing the order of displayed columns.

18 This property is only meaningful in Details view.

19 *o) AllowDrop*

20 *p) Anchor*

21 *q) AutoArrange*

22 *r) WndProc*

23  
24  
25 *Description*

1 Gets or sets whether items are automatically arranged according to the alignment property.

2 Items are also kept snapped to grid.

3 s) *BackColor*

4 t) *WndProc*

6 [C#] public override Color BackColor {get; set;}

7 [C++] public: \_\_property virtual Color get\_BackColor();public: \_\_property virtual  
8 void set\_BackColor(Color);

9 [VB] Overrides Public Property BackColor As Color

10 [JScript] public function get BackColor() : Color;public function set  
11 BackColor(Color);

13 *Description*

14 Gets or sets the background color.

15 u) *BackgroundImage*

16 v) *WndProc*

18 [C#] public override Image BackgroundImage {get; set;}

19 [C++] public: \_\_property virtual Image\* get\_BackgroundImage();public:  
20 \_\_property virtual void set\_BackgroundImage(Image\*);

21 [VB] Overrides Public Property BackgroundImage As Image

22 [JScript] public function get BackgroundImage() : Image;public function set  
23 BackgroundImage(Image);

*Description*

w) *BindingContext*

x) *BorderStyle*

y) *WndProc*

*Description*

Gets or sets the border style of the window.

z) *Bottom*

aa) *Bounds*

bb) *CanFocus*

cc) *CanSelect*

dd) *Capture*

ee) *CausesValidation*

ff) *CheckBoxes*

gg) *WndProc*

*Description*

Gets or sets whether every item will display a checkbox next to it.

The user can change the state of the item by clicking the checkbox.

**hh) CheckedIndices**

**ii) WndProc**

```
[C#] public ListView.CheckedIndexCollection CheckedIndices {get;}
[C++] public: __property ListView.CheckedIndexCollection*
get_CheckedIndices();
[VB] Public ReadOnly Property CheckedIndices As
ListView.CheckedIndexCollection
[JScript] public function get CheckedIndices() :
ListView.CheckedIndexCollection;
```

**Description**

Gets the indices of the currently checked list items.

**jj) CheckedItems**

**kk) WndProc**

```
[C#] public ListView.CheckedListViewItemCollection CheckedItems {get;}
[C++] public: __property ListView.CheckedListViewItemCollection*
get_CheckedItems();
[VB] Public ReadOnly Property CheckedItems As
ListView.CheckedListViewItemCollection
[JScript] public function get CheckedItems() :
ListView.CheckedListViewItemCollection;
```

**Description**

Gets the currently checked list items.

*ll) ClientRectangle*

*mm) ClientSize*

*nn) Columns*

*oo) WndProc*

*Description*

Gets the collection of columns.

*pp) CompanyName*

*qq) Container*

*rr) ContainsFocus*

*ss) ContextMenu*

*tt) Controls*

*uu) Created*

*vv) CreateParams*

*ww) WndProc*

*Description*

Computes the handle creation parameters for the ListView control.

xx) *Cursor*  
yy) *DataBindings*  
zz) *DefaultImeMode*  
aaa) *DefaultSize*  
bbb) *WndProc*

*Description*

Deriving classes can override this to configure a default size for their control.  
This is more efficient than setting the size in the control's constructor.

ccc) *DesignMode*  
ddd) *DisplayRectangle*  
eee) *Disposing*  
fff) *Dock*  
ggg) *Enabled*  
hhh) *Events*  
iii) *Focused*  
jjj) *FocusedItem*  
kkk) *WndProc*

*Description*

Gets the item that currently has the user focus.

This is the item that is drawn with the dotted focus rectangle around it.



1            **III)    Font**

2            **mmm) FontHeight**

3            **nnn) ForeColor**

4            **ooo) WndProc**

5  
6  
7            *Description*

8            Gets or sets the foreground color.

9            **ppp)    FullRowSelect**

10           **qqq)    WndProc**

11  
12           [C#]            public            bool            FullRowSelect            {get;            set;}

13           [C++]    public:    \_\_property    bool    get\_FullRowSelect();public:    \_\_property    void  
14           set\_FullRowSelect(bool);

15           [VB]            Public            Property            FullRowSelect            As            Boolean

16           [JScript]    public    function    get    FullRowSelect() : Boolean;public    function    set  
17           FullRowSelect(Boolean);

18  
19           *Description*

20           Gets or sets whether clicking an item will select only the item or the entire row  
21           the item is in.

22           This property is only meaningful in Details view.

*rrr) GridLines*

*sss) WndProc*

```
[C#]      public      bool      GridLines      {get;      set;}
[C++]    public:  __property  bool  get_GridLines();public:  __property  void
set_GridLines(bool);
[VB]      Public      Property      GridLines      As      Boolean
[JavaScript] public function get GridLines() : Boolean;public function set
GridLines(Boolean);
```

#### *Description*

Gets or sets whether grid lines are drawn between items and subitems.

This property is only meaningful in Details view.

*ttt) Handle*

*uuu) HasChildren*

*vvv) HeaderStyle*

*www) WndProc*

#### *Description*

Gets or sets the column header style.

This property is only meaningful in Details view.

xxx) *Height*

yyy) *HideSelection*

zzz) *WndProc*

*Description*

Gets or sets whether selected items are hidden when focus is removed from the **System.Windows.Forms.ListView** .

If false, selected items will still be highlighted (in a different color) when focus is moved away from the **System.Windows.Forms.ListView** .

aaaa) *HoverSelection*

bbbb) *WndProc*

[C#]        public        bool        HoverSelection        {get;        set;}

[C++] public: \_\_property bool get\_HoverSelection();public: \_\_property void  
set\_HoverSelection(bool);

[VB]        Public        Property        HoverSelection        As        Boolean

[JScript] public function get HoverSelection() : Boolean;public function set  
HoverSelection(Boolean);

*Description*

Gets or sets whether items can be selected by hovering over them with the mouse.

cccc) *ImeMode*

dddd) *InvokeRequired*

eeee) *IsAccessible*

ffff) *IsDisposed*

gggg) *IsHandleCreated*

hhhh) *Items*

iiii) *WndProc*

#### *Description*

Gets the list items.

jjjj) *LabelEdit*

kkkk) *WndProc*

[C#]            public            bool            LabelEdit            {get;            set;}

[C++]   public:   \_\_property   bool   get\_LabelEdit();public:   \_\_property   void  
set\_LabelEdit(bool);

[VB]            Public            Property            LabelEdit            As            Boolean

[JScript]   public   function   get   LabelEdit() : Boolean;public   function   set  
LabelEdit(Boolean);

#### *Description*

Gets or sets a value indicating whether the user can edit the labels of items in a  
**System.Windows.Forms.ListView** .

### llll) *LabelWrap*

#### mmmm) *WndProc*

```
[C#]      public      bool      LabelWrap      {get;      set;}
[C++] public: __property bool get_LabelWrap();public: __property void
set_LabelWrap(bool);
[VB]      Public      Property      LabelWrap      As      Boolean
[JavaScript] public function get LabelWrap() : Boolean;public function set
LabelWrap(Boolean);
```

#### *Description*

Gets or sets a value indicating whether item labels wrap in icon view.

### nnnn) *LargeImageList*

#### oooo) *WndProc*

```
[C#]      public      ImageList      LargeImageList      {get;      set;}
[C++] public: __property ImageList* get_LargeImageList();public: __property
void      set_LargeImageList(ImageList*);
[VB]      Public      Property      LargeImageList      As      ImageList
[JavaScript] public function get LargeImageList() : ImageList;public function set
LargeImageList(ImageList);
```

#### *Description*

Gets or sets the currently set **System.Windows.Forms.ImageList** for Large Icon mode.

1      *pppp) Left*

2      *qqqq) ListViewItemSorter*

3      *rrrr) WndProc*

4  
5  
6      *Description*

7      Gets or sets the sorting comparer for this **System.Windows.Forms.ListView** .

8      *ssss) Location*

9      *tttt) MultiSelect*

10     *uuuu) WndProc*

11  
12  
13     *Description*

14     Gets or sets a value indicating whether multiple items in a  
15     **System.Windows.Forms.ListView** control can be selected at one time.

vvvv) *Name*

www) *Parent*

xxxx) *ProductName*

yyyy) *ProductVersion*

zzzz) *RecreatingHandle*

aaaaa) *Region*

bbbbbb) *RenderRightToLeft*

ccccc) *ResizeRedraw*

dddddd) *Right*

eeeeee) *RightToLeft*

ffffff) *Scrollable*

ggggg) *WndProc*

#### *Description*

Gets or sets whether scroll bars are visible or not.

hhhhh) *SelectedIndices*

iiii) *WndProc*

```
[C#]    public    ListView.SelectedIndexCollection    SelectedIndices    {get;}
```

```
[C++]    public:    __property    ListView.SelectedIndexCollection*  
get_SelectedIndices();
```

```
[VB]    Public    ReadOnly    Property    SelectedIndices    As  
ListView.SelectedIndexCollection
```

```

1 [JScript]      public      function      get      SelectedIndices()      :
2 ListView.SelectedIndicesCollection;
3

```

#### *Description*

Gets the indices of the currently selected list of items.

*jjjjj) SelectedItems*

*kkkkk) WndProc*

```

9 [C#]  public  ListView.SelectedListViewItemCollection  SelectedItems  {get;}

```

```

10 [C++]  public:  __property  ListView.SelectedListViewItemCollection*
11 get_SelectedItems();

```

```

12 [VB]      Public      ReadOnly      Property      SelectedItems      As
13 ListView.SelectedListViewItemCollection

```

```

14 [JScript]      public      function      get      SelectedItems()      :
15 ListView.SelectedListViewItemCollection;
16

```

#### *Description*

Gets the currently selected list of items.



1        *lllll) ShowFocusCues*

2        *mmmmm)ShowKeyboardCues*

3        *nnnnn)Site*

4        *ooooo)Size*

5        *ppppp) SmallImageList*

6        *qqqqq) WndProc*

7  
8  
9        *Description*

10       Gets or sets the currently set SmallIcon image list.

11        *rrrrr) Sorting*

12        *sssss) WndProc*

13  
14       [C#]        public        SortOrder        Sorting        {get;        set;}

15       [C++]    public:    \_\_property   SortOrder   get\_Sorting();public:    \_\_property   void  
16       set\_Sorting(SortOrder);

17       [VB]        Public        Property        Sorting        As        SortOrder

18       [JScript] public    function    get    Sorting() : SortOrder;public    function    set  
19       Sorting(SortOrder);

20  
21       *Description*

22       Gets or sets the sort order of the items.



aaaaaa)Top

bbbbbb)TopItem

cccccc)WndProc

*Description*

Gets the item at the top of the list.

dddddd)TopLevelControl

eeeeee)View

ffffff) WndProc

*Description*

Gets or sets the current view.

gggggg)Visible

hhhhhh)Width

iiiiii) WindowTarget

jjjjjj) WndProc

*Description*

Occurs after a label has been edited.

**kkkkkk)WndProc**

*Description*

Occurs before a label is changed by an edit.

**lllll) WndProc**

*Description*

Occurs when the user clicks a column.

**mmmmmm)WndProc**

*Description*

Occurs when an item is activated.

**nnnnnn)WndProc**

|       |         |         |                        |                          |
|-------|---------|---------|------------------------|--------------------------|
| [C#]  | public  | event   | ItemCheckEventHandler  | ItemCheck;               |
| [C++] | public: | __event | ItemCheckEventHandler* | ItemCheck;               |
| [VB]  | Public  | Event   | ItemCheck              | As ItemCheckEventHandler |

*Description*

**ooooooo)WndProc**

|       |         |         |                       |                      |
|-------|---------|---------|-----------------------|----------------------|
| [C#]  | public  | event   | ItemDragEventHandler  | ItemDrag;            |
| [C++] | public: | __event | ItemDragEventHandler* | ItemDrag;            |
| [VB]  | Public  | Event   | ItemDrag As           | ItemDragEventHandler |

*Description*

**ppppppp)WndProc**

*Description*

**qqqqqqq)ArrangeIcons**

|           |         |          |                 |
|-----------|---------|----------|-----------------|
| [C#]      | public  | void     | ArrangeIcons(); |
| [C++]     | public: | void     | ArrangeIcons(); |
| [VB]      | Public  | Sub      | ArrangeIcons()  |
| [JScript] | public  | function | ArrangeIcons(); |

*Description*

Arranges items Large Icon or Small Icon view.

**rrrrrrr) ArrangeIcons**

|      |        |      |                                |         |
|------|--------|------|--------------------------------|---------|
| [C#] | public | void | ArrangeIcons(ListViewAlignment | value); |
|------|--------|------|--------------------------------|---------|

```

1 [C++]      public:      void      ArrangeIcons(ListViewAlignment      value);
2 [VB]      Public      Sub      ArrangeIcons(ByVal      value      As      ListViewAlignment)
3 [JScript] public function ArrangeIcons(value : ListViewAlignment); Arranges
4 items Large Icon or Small Icon view. The items are arranged according to a
5 particular                                     behavior.

```

### *Description*

In Large Icon or Small Icon view, arranges the items according to a given behavior.

The arrangement behavior may be: LVA\_DEFAULT - Aligns items according to the current alignment style LVA\_ALIGNLEFT - Aligns items along the left edge of the window LVA\_ALIGNTOP - Aligns items along the top edge of the window LVA\_SNAPTOGRID - Snaps all icons to the nearest grid position In Large Icon or Small Icon view, arranges the items according to one of the following behaviors: Arrangement behavior.

### *sssss) BeginUpdate*

```

15 [C#]      public      void      BeginUpdate();
16 [C++]      public:      void      BeginUpdate();
17 [VB]      Public      Sub      BeginUpdate()
18 [JScript]      public      function      BeginUpdate();

```

### *Description*

Prevents the **System.Windows.Forms.ListView** from redrawing itself until **System.Windows.Forms.ListView.EndUpdate** is called.

Calling this method before individually adding or removing a large number of Items will improve performance and reduce flicker on the **System.Windows.Forms.ListView** as items are being updated. Always call **System.Windows.Forms.ListView.EndUpdate** immediately after the last item is updated.

*ttttt) Clear*

|           |         |          |          |
|-----------|---------|----------|----------|
| [C#]      | public  | void     | Clear(); |
| [C++]     | public: | void     | Clear(); |
| [VB]      | Public  | Sub      | Clear()  |
| [JScript] | public  | function | Clear(); |

*Description*

Removes all items and columns from the **System.Windows.Forms.ListView** .

*uuuuuu)CreateHandle*

|           |            |           |          |                 |
|-----------|------------|-----------|----------|-----------------|
| [C#]      | protected  | override  | void     | CreateHandle(); |
| [C++]     | protected: |           | void     | CreateHandle(); |
| [VB]      | Overrides  | Protected | Sub      | CreateHandle()  |
| [JScript] | protected  | override  | function | CreateHandle(); |

*Description*

*vvvvvv)Dispose*

|           |            |           |          |                                     |
|-----------|------------|-----------|----------|-------------------------------------|
| [C#]      | protected  | override  | void     | Dispose(bool disposing);            |
| [C++]     | protected: |           | void     | Dispose(bool disposing);            |
| [VB]      | Overrides  | Protected | Sub      | Dispose(ByVal disposing As Boolean) |
| [JScript] | protected  | override  | function | Dispose(disposing : Boolean);       |

*Description*

Disposes of the component.

Call dispose when the component is no longer needed. This method removes the component from its container (if the component has a site) and triggers the dispose event.

*wwwwww)EndUpdate*

|           |         |          |              |
|-----------|---------|----------|--------------|
| [C#]      | public  | void     | EndUpdate(); |
| [C++]     | public: | void     | EndUpdate(); |
| [VB]      | Public  | Sub      | EndUpdate()  |
| [JScript] | public  | function | EndUpdate(); |

*Description*

Cancels the effect of **System.Windows.Forms.ListView.BeginUpdate** .

*xxxxxx)EnsureVisible*

|           |         |          |                     |                   |
|-----------|---------|----------|---------------------|-------------------|
| [C#]      | public  | void     | EnsureVisible(int   | index);           |
| [C++]     | public: | void     | EnsureVisible(int   | index);           |
| [VB]      | Public  | Sub      | EnsureVisible(ByVal | index As Integer) |
| [JScript] | public  | function | EnsureVisible(index | : int);           |

*Description*

Ensures that the item is visible, scrolling the view as necessary. The zero-based index of the item to scroll into view if necessary.

*yyyyyy)GetItemAt*

|      |        |              |               |    |     |     |
|------|--------|--------------|---------------|----|-----|-----|
| [C#] | public | ListViewItem | GetItemAt(int | x, | int | y); |
|------|--------|--------------|---------------|----|-----|-----|



1 [C++] public: ListViewItem\* GetItemAt(int x, int y);

2 [VB] Public Function GetItemAt(ByVal x As Integer, ByVal y As Integer) As  
3 ListViewItem

4 [JScript] public function GetItemAt(x : int, y : int) : ListViewItem;

5  
6 *Description*

7 Returns the current **System.Windows.Forms.ListViewItem** corresponding to  
8 the specific x,y coordinate.

9 *Return Value:* **System.Windows.Forms.ListViewItem** at position, or null if  
no item exists there. x position to check against y position to check against

10 *zzzzzz) GetItemRect*

11 [C#] public Rectangle GetItemRect(int index);

12 [C++] public: Rectangle GetItemRect(int index);

13 [VB] Public Function GetItemRect(ByVal index As Integer) As Rectangle

14 [JScript] public function GetItemRect(index : int) : Rectangle; Returns the  
15 specified portion of the bounding rectangle for the item of a  
16 **System.Windows.Forms.ListView**

17 .

18  
19 *Description*

20 Returns a list item's bounding rectangle, including subitems.

21 *Return Value:* Rectangle associated with the **System.Windows.Forms].**

22 **.ListViewItem' />** Returns a list item's bounding rectangle, including subitems.

23 Zero based index into ListView's list of items

24 *aaaaaaa) GetItemRect*

25 [C#] public Rectangle GetItemRect(int index, ItemBoundsPortion portion);

```

1 [C++] public: Rectangle GetItemRect(int index, ItemBoundsPortion portion);
2 [VB] Public Function GetItemRect(ByVal index As Integer, ByVal portion As
3 ItemBoundsPortion) As Rectangle
4 [JScript] public function GetItemRect(index : int, portion : ItemBoundsPortion) :
5 Rectangle;

```

### *Description*

Returns a specific portion of a list item's bounding rectangle.

*Return Value:* Rectangle associated with the

**System.Windows.Forms.ListViewItem** . Zero based index into the list of items for a **System.Windows.Forms.ListView** . The portion of the item's rectangle desired. This must be one of the values from the **System.Windows.Forms.ItemBoundsPortion** enumeration.

### *bbbbbbb)IsInputKey*

```

13
14 [C#] protected override bool IsInputKey(Keys keyData);
15 [C++] protected: bool IsInputKey(Keys keyData);
16 [VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As
17 Boolean
18 [JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

```

### *Description*

Handles special input keys, such as pgup, pgdown, home, end, and so on.

### *ccccccc)OnAfterLabelEdit*

```

23
24 [C#] protected virtual void OnAfterLabelEdit(LabelEditEventArgs e);
25 [C++] protected: virtual void OnAfterLabelEdit(LabelEditEventArgs* e);

```

[VB] Overridable Protected Sub OnAfterLabelEdit(ByVal e As  
LabelEditEventArgs)

[JScript] protected function OnAfterLabelEdit(e : LabelEditEventArgs);

*Description*

Raises the **System.Windows.Forms.ListView.AfterLabelEdit** event.  
**System.Windows.Forms.LabelEditEventArgs** object with the details

*ddddddd)OnBeforeLabelEdit*

[C#] protected virtual void OnBeforeLabelEdit(LabelEditEventArgs e);

[C++] protected: virtual void OnBeforeLabelEdit(LabelEditEventArgs\* e);

[VB] Overridable Protected Sub OnBeforeLabelEdit(ByVal e As  
LabelEditEventArgs)

[JScript] protected function OnBeforeLabelEdit(e : LabelEditEventArgs);

*Description*

Raises the **System.Windows.Forms.ListView.BeforeLabelEdit** event.  
**System.Windows.Forms.LabelEditEventArgs** object with the details

*eeeeeee)OnColumnClick*

[C#] protected virtual void OnColumnClick(ColumnClickEventArgs e);

[C++] protected: virtual void OnColumnClick(ColumnClickEventArgs\* e);

[VB] Overridable Protected Sub OnColumnClick(ByVal e As  
ColumnClickEventArgs)

[JScript] protected function OnColumnClick(e : ColumnClickEventArgs);

*Description*

Raises the **System.Windows.Forms.ListView.ColumnClick** event.  
**System.Windows.Forms.ColumnClickEventArgs** object with the details.

*fffffff) OnEnabledChanged*

```
[C#]    protected    override    void    OnEnabledChanged(EventArgs    e);
[C++]    protected:    void    OnEnabledChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnEnabledChanged(ByVal e As EventArgs)
[JScript] protected override function OnEnabledChanged(e : EventArgs);
```

*Description*

Raises the EnabledChanged event. The EventArgs that contains the event data.

*ggggggg)OnFontChanged*

```
[C#]    protected    override    void    OnFontChanged(EventArgs    e);
[C++]    protected:    void    OnFontChanged(EventArgs*    e);
[VB]    Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)
[JScript] protected override function OnFontChanged(e : EventArgs);
```

*Description*

Raises the FontChanged event. The EventArgs that contains the event data.

*hhhhhhh)OnHandleCreated*

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
```

1 [C++] protected: void OnHandleCreated(EventArgs\* e);

2 [VB] Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)

3 [JScript] protected override function OnHandleCreated(e : EventArgs);

4  
5 *Description*

6 **iiiiii) OnHandleDestroyed**

7  
8 [C#] protected override void OnHandleDestroyed(EventArgs e);

9 [C++] protected: void OnHandleDestroyed(EventArgs\* e);

10 [VB] Overrides Protected Sub OnHandleDestroyed(ByVal e As EventArgs)

11 [JScript] protected override function OnHandleDestroyed(e : EventArgs);

12  
13 *Description*

14  
15 **jjjjjj) OnItemActivate**

16  
17 [C#] protected virtual void OnItemActivate(EventArgs e);

18 [C++] protected: virtual void OnItemActivate(EventArgs\* e);

19 [VB] Overridable Protected Sub OnItemActivate(ByVal e As EventArgs)

20 [JScript] protected function OnItemActivate(e : EventArgs);

21  
22 *Description*

23 Raises the **System.Windows.Forms.ListView.ItemActivate** event. Event  
24 object.  
25

### *kkkkkkk)OnItemCheck*

```
1
2
3 [C#] protected virtual void OnItemCheck(ItemCheckEventArgs ice);
4 [C++] protected: virtual void OnItemCheck(ItemCheckEventArgs* ice);
5 [VB] Overridable Protected Sub OnItemCheck(ByVal ice As
6 ItemCheckEventArgs)
7 [JScript] protected function OnItemCheck(ice : ItemCheckEventArgs);
8
```

#### *Description*

This is the code that actually fires the KeyEventArgs. Don't forget to call base.OnItemCheck() to ensure that itemCheck vents are correctly fired for all other keys. The KeyPress event that was fired.

### *lllllll) OnItemDrag*

```
12
13
14 [C#] protected virtual void OnItemDrag(ItemDragEventArgs e);
15 [C++] protected: virtual void OnItemDrag(ItemDragEventArgs* e);
16 [VB] Overridable Protected Sub OnItemDrag(ByVal e As ItemDragEventArgs)
17 [JScript] protected function OnItemDrag(e : ItemDragEventArgs);
18
```

#### *Description*

Raises the **System.Windows.Forms.ListView.ItemDrag** event.

### *mmmmmmm)OnSelectedIndexChanged*

```
21
22
23 [C#] protected virtual void OnSelectedIndexChanged(EventArgs e);
24 [C++] protected: virtual void OnSelectedIndexChanged(EventArgs* e);
25 [VB] Overridable Protected Sub OnSelectedIndexChanged(ByVal e As
```

EventArgs)

[JScript] protected function OnSelectedIndexChanged(e : EventArgs);

*Description*

Raises the **System.Windows.Forms.ListView.SelectedIndexChanged** event. Event object with the details

*nnnnnnn)RealizeProperties*

[C#] protected void RealizeProperties();

[C++] protected: void RealizeProperties();

[VB] Protected Sub RealizeProperties()

[JScript] protected function RealizeProperties();

*Description*

*oooooooo)Sort*

[C#] public void Sort();

[C++] public: void Sort();

[VB] Public Sub Sort()

[JScript] public function Sort();

*Description*

Updated the sorted order Updated the sorted order

**ppppppp)ToString**

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

*Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

**qqqqqqq)UpdateExtendedStyles**

|           |            |          |                         |
|-----------|------------|----------|-------------------------|
| [C#]      | protected  | void     | UpdateExtendedStyles(); |
| [C++]     | protected: | void     | UpdateExtendedStyles(); |
| [VB]      | Protected  | Sub      | UpdateExtendedStyles()  |
| [JScript] | protected  | function | UpdateExtendedStyles(); |

*Description*

**rrrrrrr)WndProc**

|           |            |           |                      |                             |
|-----------|------------|-----------|----------------------|-----------------------------|
| [C#]      | protected  | override  | void                 | WndProc(ref Message m);     |
| [C++]     | protected: | void      | WndProc(Message* m); |                             |
| [VB]      | Overrides  | Protected | Sub                  | WndProc(ByRef m As Message) |
| [JScript] | protected  | override  | function             | WndProc(m : Message);       |



*Description*

ListViewAlignment enumeration (System.Windows.Forms)

*a) WndProc*

*Description*

Specifies how items align in the **System.Windows.Forms.ListView** .

Use the members of this enumeration to set the value of the **System.Windows.Forms.ListView.Alignment** property and the **System.Windows.Forms.ListView.ArrangeIcons(System.Windows.Forms.ListViewAlignment)** method of the **System.Windows.Forms.ListView** control.

*b) WndProc*

|           |         |       |                   |                    |
|-----------|---------|-------|-------------------|--------------------|
| [C#]      | public  | const | ListViewAlignment | Default;           |
| [C++]     | public: | const | ListViewAlignment | Default;           |
| [VB]      | Public  | Const | Default As        | ListViewAlignment  |
| [JScript] | public  | var   | Default :         | ListViewAlignment; |

*Description*

When the user moves an item, it remains where it is dropped.

*c) WndProc*

|       |         |       |                   |       |
|-------|---------|-------|-------------------|-------|
| [C#]  | public  | const | ListViewAlignment | Left; |
| [C++] | public: | const | ListViewAlignment | Left; |

```

1 [VB]      Public      Const      Left      As      ListViewAlignment
2 [JScript]      public      var      Left      :      ListViewAlignment;

```

#### *Description*

Items are aligned to the left of the **System.Windows.Forms.ListView** control.

#### *d) WndProc*

```

8 [C#]      public      const      ListViewAlignment      SnapToGrid;
9 [C++]      public:      const      ListViewAlignment      SnapToGrid;
10 [VB]      Public      Const      SnapToGrid      As      ListViewAlignment
11 [JScript]      public      var      SnapToGrid      :      ListViewAlignment;

```

#### *Description*

Items are aligned to an invisible grid in the control. When the user moves an item, it moves to the closest juncture in the grid.

#### *e) WndProc*

```

17 [C#]      public      const      ListViewAlignment      Top;
18 [C++]      public:      const      ListViewAlignment      Top;
19 [VB]      Public      Const      Top      As      ListViewAlignment
20 [JScript]      public      var      Top      :      ListViewAlignment;

```

#### *Description*

Items are aligned to the top of the **System.Windows.Forms.ListView** control.

1 ListViewItem class (System.Windows.Forms)

2 a) ToString

3  
4  
5 Description

6 Implements an item of a **System.Windows.Forms.ListView** .

7 b) ListViewItem

8 Example Syntax:

9 c) ToString

10  
11 [C#] public ListViewItem();  
12 [C++] public: ListViewItem();  
13 [VB] Public Sub New()  
14 [JScript] public function ListViewItem();  
15

16 Description

17  
18 d) ListViewItem

19 Example Syntax:

20 e) ToString

21  
22 [C#] public ListViewItem(string text);  
23 [C++] public: ListViewItem(String\* text);  
24 [VB] Public Sub New(ByVal text As String)  
25

1 [JScript] public function ListViewItem(text : String);

3 *Description*

5 *f) ListViewItem*

6 *Example Syntax:*

7 *g) ToString*

9 [C#] public ListViewItem(string[] items);

10 [C++] public: ListViewItem(String\* items \_\_gc[]);

11 [VB] Public Sub New(ByVal items() As String)

12 [JScript] public function ListViewItem(items : String[]);

14 *Description*

16 *h) ListViewItem*

17 *Example Syntax:*

18 *i) ToString*

20 [C#] public ListViewItem(ListViewItem.ListViewSubItem[] subItems, int  
21 imageIndex);

22 [C++] public: ListViewItem(ListViewItem.ListViewSubItem\* subItems[], int  
23 imageIndex);

24 [VB] Public Sub New(ByVal subItems() As ListViewItem.ListViewSubItem,

```

1 ByVal          imageIndex          As          Integer)
2 [JScript]      public      function      ListViewItem(subItems      :
3 ListViewItem.ListViewSubItem[],      imageIndex      :      int);
4

```

#### *Description*

#### *j) ListViewItem*

#### *Example Syntax:*

#### *k) ToString*

```

11 [C#]      public      ListViewItem(string      text,      int      imageIndex);
12 [C++]      public:      ListViewItem(String*      text,      int      imageIndex);
13 [VB] Public Sub New(ByVal text As String, ByVal imageIndex As Integer)
14 [JScript] public function ListViewItem(text : String, imageIndex : int);
15

```

#### *Description*

#### *l) ListViewItem*

#### *Example Syntax:*

#### *m) ToString*

```

22 [C#]      public      ListViewItem(string[]      items,      int      imageIndex);
23 [C++]      public:      ListViewItem(String*      items      __gc[],      int      imageIndex);
24 [VB] Public Sub New(ByVal items() As String, ByVal imageIndex As Integer)
25

```

1 [JScript] public function ListViewItem(items : String[], imageIndex : int);

2  
3 *Description*

4  
5 *n) ListViewItem*

6 *Example Syntax:*

7 *o) ToString*

8  
9 [C#] public ListViewItem(string[] items, int imageIndex, Color foreColor, Color  
10 backColor, Font font);

11 [C++] public: ListViewItem(String\* items \_\_gc[], int imageIndex, Color  
12 foreColor, Color backColor, Font\* font);

13 [VB] Public Sub New(ByVal items() As String, ByVal imageIndex As Integer,  
14 ByVal foreColor As Color, ByVal backColor As Color, ByVal font As Font)

15 [JScript] public function ListViewItem(items : String[], imageIndex : int,  
16 foreColor : Color, backColor : Color, font : Font);

17  
18 *Description*

19  
20 *p) BackColor*

21 *q) ToString*

22  
23 [C#] public Color BackColor {get; set;}

24 [C++] public: \_\_property Color get\_BackColor();public: \_\_property void

1 set\_BackColor(Color);

2 [VB] Public Property BackColor As Color

3 [JScript] public function get BackColor() : Color;public function set

4 BackColor(Color);

5  
6 *Description*

7 The font that this item will be displayed in. If its value is null, it will be displayed  
8 using the global font for the **System.Windows.Forms.ListView** control that  
hosts it.

9 *r) Bounds*

10 *s) ToString*

11  
12 [C#] public Rectangle Bounds {get;}

13 [C++] public: \_\_property Rectangle get\_Bounds();

14 [VB] Public ReadOnly Property Bounds As Rectangle

15 [JScript] public function get Bounds() : Rectangle;

16  
17 *Description*

18 Returns the bounding rectangle of the  
19 **System.Windows.Forms.ListViewItem** , including subitems.

20 *t) Checked*

21 *u) ToString*

22  
23 [C#] public bool Checked {get; set;}

24 [C++] public: \_\_property bool get\_Checked();public: \_\_property void

25 set\_Checked(bool);

```

1 [VB]      Public      Property      Checked      As      Boolean
2 [JScript] public function get Checked() : Boolean;public function set
3 Checked(Boolean);
4

```

#### *Description*

v) *Focused*

w) *ToString*

```

10 [C#]      public      bool      Focused      {get;      set;}
11

```

```

12 [C++] public: __property bool get_Focused();public: __property void
13 set_Focused(bool);
14

```

```

15 [VB]      Public      Property      Focused      As      Boolean
16

```

```

17 [JScript] public function get Focused() : Boolean;public function set
18 Focused(Boolean);
19

```

#### *Description*

Returns the focus state of the **System.Windows.Forms.ListViewItem** .

x) *Font*

y) *ToString*

```

22 [C#]      public      Font      Font      {get;      set;}
23

```

```

24 [C++] public: __property Font* get_Font();public: __property void
25 set_Font(Font*);

```



1 [VB]           Public           Property           Font           As           Font

2 [JScript] public function get Font() : Font;public function set Font(Font);

4 *Description*

6           z)       ***ForeColor***

7           aa)     ***ToString***

9 [C#]           public           Color           ForeColor           {get;           set;}

10 [C++] public: \_\_property Color get\_ForeColor();public: \_\_property void  
11 set\_ForeColor(Color);

12 [VB]           Public           Property           ForeColor           As           Color

13 [JScript] public function get ForeColor() : Color;public function set  
14 ForeColor(Color);

16 *Description*

18           bb)     ***ImageIndex***

19           cc)     ***ToString***

21 [C#]           public           int           ImageIndex           {get;           set;}

22 [C++] public: \_\_property int get\_ImageIndex();public: \_\_property void  
23 set\_ImageIndex(int);

24 [VB]           Public           Property           ImageIndex           As           Integer

```

1 [JScript] public function get ImageIndex() : int;public function set
2 ImageIndex(int);
3

```

#### *Description*

Returns the current **System.Windows.Forms.ImageList** .

*dd) ImageList*

*ee) ToString*

```

9 [C#]          public          ImageList          ImageList          {get;}
10 [C++]         public:         __property         ImageList*         get_ImageList();
11 [VB]          Public          ReadOnly          Property          ImageList          As          ImageList
12 [JScript]     public          function          get          ImageList()          :          ImageList;
13

```

#### *Description*

*ff) Index*

*gg) ToString*

```

19 [C#]          public          int          Index          {get;}
20 [C++]         public:         __property         int          get_Index();
21 [VB]          Public          ReadOnly          Property          Index          As          Integer
22 [JScript]     public          function          get          Index()          :          int;
23

```

#### *Description*

Returns the index in the **System.Windows.Forms.ListView** control for the **System.Windows.Forms.ListViewItem** .

*hh) ListView*

*ii) ToString*

[C#]            public            ListView            ListView            {get;}

[C++]           public:            \_\_property            ListView\*            get\_ListView();

[VB]           Public            ReadOnly            Property            ListView            As            ListView

[JScript]       public            function            get            ListView()            :            ListView;

#### *Description*

Returns the **System.Windows.Forms.ListView** control that holds this **System.Windows.Forms.ListViewItem** .

*jj) Selected*

*kk) ToString*

[C#]            public            bool            Selected            {get;            set;}

[C++]           public:            \_\_property            bool            get\_Selected();public:            \_\_property            void  
set\_Selected(bool);

[VB]           Public            Property            Selected            As            Boolean

[JScript]       public            function            get            Selected()            :            Boolean;public            function            set  
Selected(Boolean);

#### *Description*

Gets or set a value indicating whether the item is selected within a **System.Windows.Forms.ListView** control.

1        **ll)     StateImageIndex**

2        **mm)   ToString**

3  
4        [C#]        public        int        StateImageIndex        {get;        set;}

5        [C++]        public:   \_\_property   int   get\_StateImageIndex();public:   \_\_property   void  
6        set\_StateImageIndex(int);

7        [VB]        Public        Property        StateImageIndex        As        Integer

8        [JScript]        public        function        get        StateImageIndex()        :        int;public        function        set  
9        StateImageIndex(int);

10  
11        *Description*

12  
13        **nn)     SubItems**

14        **oo)     ToString**

15  
16        [C#]        public        ListViewItem.ListViewSubItemCollection        SubItems        {get;}

17        [C++]        public:        \_\_property        ListViewItem.ListViewSubItemCollection\*  
18        get\_SubItems();

19        [VB]        Public        ReadOnly        Property        SubItems        As  
20        ListViewItem.ListViewSubItemCollection

21        [JScript]        public        function        get        SubItems()        :  
22        ListViewItem.ListViewSubItemCollection;

23  
24        *Description*

*pp) Tag*

*qq) ToString*

```
[C#]          public          object          Tag          {get;          set;}
[C++] public: __property Object* get_Tag();public: __property void
set_Tag(Object*);
[VB]          Public          Property          Tag          As          Object
[JScript] public function get Tag() : Object;public function set Tag(Object);
```

*Description*

*rr) Text*

*ss) ToString*

```
[C#]          public          string          Text          {get;          set;}
[C++] public: __property String* get_Text();public: __property void
set_Text(String*);
[VB]          Public          Property          Text          As          String
[JScript] public function get Text() : String;public function set Text(String);
```

*Description*

Text associated with this **System.Windows.Forms.ListViewItem** .

**tt) *UseItemStyleForSubItems***

### *uu) ToString*

```
[C#] public bool UseItemStyleForSubItems {get; set;}
```

```
[C++] public: __property bool get_UseItemStyleForSubItems();public: __property
void set_UseItemStyleForSubItems(bool);
```

|      |        |          |                         |    |         |
|------|--------|----------|-------------------------|----|---------|
| [VB] | Public | Property | UseItemStyleForSubItems | As | Boolean |
|------|--------|----------|-------------------------|----|---------|

```
[JScript] public function get UseItemStyleForSubItems() : Boolean;public
function set UseItemStyleForSubItems(Boolean);
```

*Description*

Gets or sets a value indicating whether the font and coloring for the **System.Windows.Forms.ListViewItem** will be used for all of its subitems.

vv) *BeginEdit*

```
[C#]      public      void      BeginEdit();
```

```
[C++]      public:      void      BeginEdit();
```

[VB]                      Public                      Sub                      BeginEdit()

```
[JScript]      public      function      BeginEdit();
```

### Description

Begins the editing of the item's label.

Only effective if the **System.Windows.Forms.ListView.LabelEdit** property is **true** .

ww) *Clone*

```
1
2 [C#]          public          virtual          object          Clone();
3 [C++]          public:          virtual          Object*          Clone();
4 [VB]  Overridable  Public  Function  Clone()  As  Object
5 [JScript]      public          function          Clone()          :          Object;
```

8 *Description*

xx) *Deserialize*

```
11 [C#] protected virtual void Deserialize(SerializationInfo info, StreamingContext
12 context);
13 [C++] protected: virtual void Deserialize(SerializationInfo* info,
14 StreamingContext context);
15 [VB] Overridable Protected Sub Deserialize(ByVal info As SerializationInfo,
16 ByVal context As StreamingContext)
17 [JScript] protected function Deserialize(info : SerializationInfo, context :
18 StreamingContext);
19
```

21 *Description*

yy) *EnsureVisible*

```
24 [C#]          public          virtual          void  *          EnsureVisible();
25
```

|           |             |          |      |                  |
|-----------|-------------|----------|------|------------------|
| [C++]     | public:     | virtual  | void | EnsureVisible(); |
| [VB]      | Overridable | Public   | Sub  | EnsureVisible()  |
| [JScript] | public      | function |      | EnsureVisible(); |

#### Description

Ensures that the item is visible, scrolling the view as necessary.

#### zz) GetBounds

|           |         |           |                                               |              |
|-----------|---------|-----------|-----------------------------------------------|--------------|
| [C#]      | public  | Rectangle | GetBounds(ItemBoundsPortion                   | portion);    |
| [C++]     | public: | Rectangle | GetBounds(ItemBoundsPortion                   | portion);    |
| [VB]      | Public  | Function  | GetBounds(ByVal portion As ItemBoundsPortion) | As Rectangle |
| [JScript] | public  | function  | GetBounds(portion : ItemBoundsPortion)        | : Rectangle; |

#### Description

Returns a specific portion of the bounding rectangle for the **System.Windows.Forms.ListViewItem**.

*Return Value:* Rectangle associated with the **System.Windows.Forms.ListViewItem**. An **System.Windows.Forms.ItemBoundsPortion** that represents a portion of the **System.Windows.Forms.ListViewItem** for which to retrieve the bounding rectangle.

#### aaa) Remove

|       |             |         |      |           |
|-------|-------------|---------|------|-----------|
| [C#]  | public      | virtual | void | Remove(); |
| [C++] | public:     | virtual | void | Remove(); |
| [VB]  | Overridable | Public  | Sub  | Remove()  |



1 [JScript] public function Remove();

3 *Description*

5 *bbb) Serialize*

7 [C#] protected virtual void Serialize(SerializationInfo info, StreamingContext  
8 context);

9 [C++] protected: virtual void Serialize(SerializationInfo\* info, StreamingContext  
10 context);

11 [VB] Overridable Protected Sub Serialize(ByVal info As SerializationInfo, ByVal  
12 context As StreamingContext)

13 [JScript] protected function Serialize(info : SerializationInfo, context :  
14 StreamingContext);

16 *Description*

17 Saves this **System.Windows.Forms.ListViewItem** object to the given data  
stream.

18 *ccc) ISerializable.GetObjectData*

20  
21 [C#] void ISerializable.GetObjectData(SerializationInfo info, StreamingContext  
22 context);

23 [C++] void ISerializable::GetObjectData(SerializationInfo\* info,  
24 StreamingContext context);

25 [VB] Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As

```

1 StreamingContext)           Implements           ISerializable.GetObjectData
2 [JScript] function ISerializable.GetObjectData(info : SerializationInfo, context :
3 StreamingContext);
4         ddd) ToString
5
6 [C#]           public           override           string           ToString();
7 [C++]           public:           String*           ToString();
8 [VB]   Overrides   Public   Function   ToString()   As   String
9 [JScript]   public   override   function   ToString()   :   String;

```

#### *Description*

ListView.ListViewItemCollection class (System.Windows.Forms)

#### *a) ToString*

#### *Description*

#### *b) ListView.ListViewItemCollection*

*Example Syntax:*

#### *c) ToString*

```

23 [C#]           public           ListView.ListViewItemCollection(ListView   owner);
24 [C++]           public:           ListViewItemCollection(ListView*   owner);

```

1 [VB] Public Sub New(ByVal owner As ListView)

2 [JScript] public function ListView.ListViewItemCollection(owner : ListView);

4 *Description*

6 *d) Count*

7 *e) ToString*

9 [C#] public int Count {get;}

10 [C++] public: \_\_property int get\_Count();

11 [VB] Public ReadOnly Property Count As Integer

12 [JScript] public function get Count() : int;

14 *Description*

15 Returns the total number of items within the list view.

16 *Return Value:* Integer count of items.

17 *f) IsReadOnly*

18 *g) ToString*

20 [C#] public bool IsReadOnly {get;}

21 [C++] public: \_\_property bool get\_IsReadOnly();

22 [VB] Public ReadOnly Property IsReadOnly As Boolean

23 [JScript] public function get IsReadOnly() : Boolean;

25 *Description*

h) *Item*

i) *ToString*

[C#] public virtual ListViewItem this[int displayIndex] {get; set;}

[C++] public: \_\_property virtual ListViewItem\* get\_Item(int displayIndex); public: \_\_property virtual void set\_Item(int displayIndex, ListViewItem\*);

[VB] Overridable Public Default Property Item(ByVal displayIndex As Integer) As ListViewItem

[JScript] returnValue = ListViewItemCollectionObject.Item(displayIndex); ListViewItemCollectionObject.Item(displayIndex) = returnValue;

### *Description*

Returns a **System.Windows.Forms.ListViewItem** given it's zero-based index into the **System.Windows.Forms.ListView** .

j) *Add*

[C#] public virtual ListViewItem Add(ListViewItem value);

[C++] public: virtual ListViewItem\* Add(ListViewItem\* value);

[VB] Overridable Public Function Add(ByVal value As ListViewItem) As ListViewItem

[JScript] public function Add(value : ListViewItem) : ListViewItem;

## Description

Add an item to the **System.Windows.Forms.ListView** . The item will be inserted either in the correct sorted position, or, if no sorting is set, at the end of the list.

*Return Value:* The **System.Windows.Forms.ListViewItem** object passed in. The **System.Windows.Forms.ListViewItem** object containing the item data

### k) Add

[C#] public virtual ListViewItem Add(string text);

[C++] public: virtual ListViewItem\* Add(String\* text);

[VB] Overridable Public Function Add(ByVal text As String) As ListViewItem

[JScript] public function Add(text : String) : ListViewItem;

## Description

Add an item to the **System.Windows.Forms.ListView** . The item will be inserted either in the correct sorted position, or, if no sorting is set, at the end of the list.

*Return Value:* A **System.Windows.Forms.ListViewItem** object representing the added item Add an item to the ListView. The item will be inserted either in the correct sorted position, or, if no sorting is set, at the end of the list. The text of the item.

### l) Add

[C#] public virtual ListViewItem Add(string text, int imageIndex);

[C++] public: virtual ListViewItem\* Add(String\* text, int imageIndex);

[VB] Overridable Public Function Add(ByVal text As String, ByVal imageIndex

As Integer) As ListViewItem

[JScript] public function Add(text : String, imageIndex : int) : ListViewItem;

*Description*

Add an item to the **System.Windows.Forms.ListView** . The item will be inserted either in the correct sorted position, or, if no sorting is set, at the end of the list.

*Return Value:* A **System.Windows.Forms.ListViewItem** object representing the added item Add an item to the ListView. The item will be inserted either in the correct sorted position, or, if no sorting is set, at the end of the list. The text of the item. An index of the image in the **System.Windows.Forms.ImageList** object assigned to the **System.Windows.Forms.ListView.ImageList** property of the **System.Windows.Forms.ListView** to associate with the item.

*m) AddRange*

|           |         |          |                         |                           |
|-----------|---------|----------|-------------------------|---------------------------|
| [C#]      | public  | void     | AddRange(ListViewItem[] | values);                  |
| [C++]     | public: | void     | AddRange(ListViewItem*  | values[]);                |
| [VB]      | Public  | Sub      | AddRange(ByVal          | values() As ListViewItem) |
| [JScript] | public  | function | AddRange(values         | : ListViewItem[]);        |

*Description*

*n) Clear*

|           |             |          |      |          |
|-----------|-------------|----------|------|----------|
| [C#]      | public      | virtual  | void | Clear(); |
| [C++]     | public:     | virtual  | void | Clear(); |
| [VB]      | Overridable | Public   | Sub  | Clear()  |
| [JScript] | public      | function |      | Clear(); |

*Description*

Removes all items from the list view.

*o) Contains*

```
[C#]      public      bool      Contains(ListViewItem      item);
[C++]      public:      bool      Contains(ListViewItem*      item);
[VB] Public Function Contains(ByVal item As ListViewItem) As Boolean
[JScript] public function Contains(item : ListViewItem) : Boolean;
```

*Description*

*p) CopyTo*

```
[C#]      public      void      CopyTo(Array      dest,      int      index);
[C++]      public:      __sealed      void      CopyTo(Array*      dest,      int      index);
[VB] NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As
Integer)
[JScript] public function CopyTo(dest : Array, index : int);
```

*Description*

*q) GetEnumerator*

```
[C#]      public      IEnumerator      GetEnumerator();
[C++]      public:      __sealed      IEnumerator*      GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
```

1 [JScript] public function GetEnumerator() : IEnumerator;

3 *Description*

5 *r) IndexOf*

7 [C#] public int IndexOf(ListViewItem item);

8 [C++] public: int IndexOf(ListViewItem\* item);

9 [VB] Public Function IndexOf(ByVal item As ListViewItem) As Integer

10 [JScript] public function IndexOf(item : ListViewItem) : int;

12 *Description*

14 *s) Insert*

16 [C#] public ListViewItem Insert(int index, ListViewItem item);

17 [C++] public: ListViewItem\* Insert(int index, ListViewItem\* item);

18 [VB] Public Function Insert(ByVal index As Integer, ByVal item As

19 ListViewItem) As ListViewItem

20 [JScript] public function Insert(index : int, item : ListViewItem) : ListViewItem;

22 *Description*



**t) Insert**

```
[C#]    public    ListViewItem    Insert(int    index,    string    text);
[C++]   public:   ListViewItem*    Insert(int    index,    String*    text);
[VB]    Public Function Insert(ByVal index As Integer, ByVal text As String) As
ListViewItem
[JScript] public function Insert(index : int, text : String) : ListViewItem;
```

*Description*

**u) Insert**

```
[C#]    public ListViewItem Insert(int index, string text, int imageIndex);
[C++]   public: ListViewItem* Insert(int index, String* text, int imageIndex);
[VB]    Public Function Insert(ByVal index As Integer, ByVal text As String, ByVal
imageIndex          As          Integer)          As          ListViewItem
[JScript] public function Insert(index : int, text : String, imageIndex : int) :
ListViewItem;
```

*Description*

**v) Remove**

```
[C#]    public    virtual    void    Remove(ListViewItem    item);
[C++]   public:   virtual    void    Remove(ListViewItem*    item);
```

```

1 [VB] Overridable Public Sub Remove(ByVal item As ListViewItem)
2 [JScript] public function Remove(item : ListViewItem);

```

#### *Description*

Removes an item from the **System.Windows.Forms.ListView** .  
**System.Windows.Forms.ListViewItem** to be removed

#### *w) RemoveAt*

```

8 [C#] public virtual void RemoveAt(int index);
9 [C++] public: virtual void RemoveAt(int index);
10 [VB] Overridable Public Sub RemoveAt(ByVal index As Integer)
11 [JScript] public function RemoveAt(index : int);

```

#### *Description*

Removes an item from the **System.Windows.Forms.ListView** . The zero-based index of the item that is to be removed.

#### *x) IList.Add*

```

18 [C#] int IList.Add(object item);
19 [C++] int IList::Add(Object* item);
20 [VB] Function Add(ByVal item As Object) As Integer Implements IList.Add
21 [JScript] function IList.Add(item : Object) : int;

```

#### *y) IList.Contains*

```

24 [C#] bool IList.Contains(object item);

```

1 [C++] bool IList::Contains(Object\* item);

2 [VB] Function Contains(ByVal item As Object) As Boolean Implements

3 IList.Contains

4 [JScript] function IList.Contains(item : Object) : Boolean;

5 z) *IList.IndexOf*

7 [C#] int IList.IndexOf(object item);

8 [C++] int IList::IndexOf(Object\* item);

9 [VB] Function IndexOf(ByVal item As Object) As Integer Implements

10 IList.IndexOf

11 [JScript] function IList.IndexOf(item : Object) : int;

12 aa) *IList.Insert*

14 [C#] void IList.Insert(int index, object item);

15 [C++] void IList::Insert(int index, Object\* item);

16 [VB] Sub Insert(ByVal index As Integer, ByVal item As Object) Implements

17 IList.Insert

18 [JScript] function IList.Insert(index : int, item : Object);

19 bb) *IList.Remove*

21 [C#] void IList.Remove(object item);

22 [C++] void IList::Remove(Object\* item);

23 [VB] Sub Remove(ByVal item As Object) Implements IList.Remove

24 [JScript] function IList.Remove(item : Object);

ListViewItemConverter class (System.Windows.Forms)

a) *ToString*

*Description*

**System.Windows.Forms.ListViewItemConverter** is a class that can be used to convert **System.Windows.Forms.ListViewItem** objects from one data type to another.

b) *ListViewItemConverter*

*Example Syntax:*

c) *ToString*

[C#] public ListViewItemConverter();

[C++] public: ListViewItemConverter();

[VB] Public Sub New()

[JScript] public function ListViewItemConverter();

d) *CanConvertTo*

[C#] public override bool CanConvertTo(ITypeDescriptorContext context, Type destinationType);

[C++] public: bool CanConvertTo(ITypeDescriptorContext\* context, Type\* destinationType);

[VB] Overrides Public Function CanConvertTo(ByVal context As ITypeDescriptorContext, ByVal destinationType As Type) As Boolean

[JScript] public override function CanConvertTo(context :

1 ITypeDescriptorContext, destinationType : Type) : Boolean;

3 *Description*

4 Gets a value indicating whether this converter can convert an object to the given destination type using the context.

5 *Return Value:* **true** if this converter can perform the conversion; otherwise, **false** .

6 The *context* parameter can be used to extract additional information about the environment this converter is being invoked from. This can be **null** , so always check. Also, properties on the context object can return **null** . An **System.ComponentModel.ITypeDescriptorContext** that provides a format context. A **System.Type** that represents the type you wish to convert to.

10 e) *ConvertTo*

12 [C#] public override object ConvertTo(ITypeDescriptorContext context,

13 CultureInfo culture, object value, Type destinationType);

14 [C++] public: Object\* ConvertTo(ITypeDescriptorContext\* context, CultureInfo\*

15 culture, Object\* value, Type\* destinationType);

16 [VB] Overrides Public Function ConvertTo(ByVal context As

17 ITypeDescriptorContext, ByVal culture As CultureInfo, ByVal value As Object,

18 ByVal destinationType As Type) As Object

19 [JScript] public override function ConvertTo(context : ITypeDescriptorContext,

20 culture : CultureInfo, value : Object, destinationType : Type) : Object;

22 *Description*

23 Converts the given object to another type.

24 *Return Value:* The converted object.

25 The most common types to convert are to and from a string object. The default implementation will make a call to **System.Object.ToString** on the object if

the object is valid and if the destination type is string. If this cannot convert to the destination type, this will throw a **System.NotSupportedException**. A formatter context. This object can be used to extract additional information about the environment this converter is being invoked from. This may be null, so you should always check. Also, properties on the context object may also return null. An optional culture info. If not supplied the current culture is assumed. The object to convert. The type to convert the object to.

ListViewItem.ListViewSubItem class (System.Windows.Forms)

*a) ToString*

*Description*

*b) ListViewItem.ListViewSubItem*

*Example Syntax:*

*c) ToString*

|           |         |                                          |
|-----------|---------|------------------------------------------|
| [C#]      | public  | ListViewItem.ListViewSubItem();          |
| [C++]     | public: | ListViewSubItem();                       |
| [VB]      | Public  | Sub New()                                |
| [JScript] | public  | function ListViewItem.ListViewSubItem(); |

*Description*

*d) ListViewItem.ListViewSubItem*

*Example Syntax:*

e) *ToString*

```
[C#] public ListViewItem.ListViewSubItem(ListViewItem owner, string text);  
[C++] public: ListViewItem(ListViewItem* owner, String* text);  
[VB] Public Sub New(ByVal owner As ListViewItem, ByVal text As String)  
[JScript] public function ListViewItem.ListViewSubItem(owner : ListViewItem,  
text : String);
```

*Description*

f) *ListViewItem.ListViewSubItem*

*Example Syntax:*

g) *ToString*

```
[C#] public ListViewItem.ListViewSubItem(ListViewItem owner, string text,  
Color foreColor, Color backColor, Font font);  
[C++] public: ListViewItem(ListViewItem* owner, String* text, Color  
foreColor, Color backColor, Font* font);  
[VB] Public Sub New(ByVal owner As ListViewItem, ByVal text As String,  
ByVal foreColor As Color, ByVal backColor As Color, ByVal font As Font)  
[JScript] public function ListViewItem.ListViewSubItem(owner : ListViewItem,  
text : String, foreColor : Color, backColor : Color, font : Font);
```

*Description*

h) *BackColor*

i) *ToString*

[C#] public Color BackColor {get; set;}

[C++] public: \_\_property Color get\_BackColor();public: \_\_property void  
set\_BackColor(Color);

[VB] Public Property BackColor As Color

[JScript] public function get BackColor() : Color;public function set  
BackColor(Color);

*Description*

j) *Font*

k) *ToString*

[C#] public Font Font {get; set;}

[C++] public: \_\_property Font\* get\_Font();public: \_\_property void  
set\_Font(Font\*);

[VB] Public Property Font As Font

[JScript] public function get Font() : Font;public function set Font(Font);

*Description*



1            ***l)      ForeColor***

2            ***m)      ToString***

3  
4    [C#]            public            Color            ForeColor            {get;            set;}

5    [C++]   public:   \_\_property   Color   get\_ForeColor();public:   \_\_property   void  
6    set\_ForeColor(Color);

7    [VB]            Public            Property            ForeColor            As            Color

8    [JScript]   public   function   get   ForeColor()   :   Color;public   function   set  
9    ForeColor(Color);

10  
11    *Description*

12  
13            ***n)      Text***

14            ***o)      ToString***

15  
16    [C#]            public            string            Text            {get;            set;}

17    [C++]   public:   \_\_property   String\*   get\_Text();public:   \_\_property   void  
18    set\_Text(String\*);

19    [VB]            Public            Property            Text            As            String

20    [JScript]   public   function   get   Text()   :   String;public   function   set   Text(String);

21  
22    *Description*

**p) ResetStyle**

|           |         |          |               |
|-----------|---------|----------|---------------|
| [C#]      | public  | void     | ResetStyle(); |
| [C++]     | public: | void     | ResetStyle(); |
| [VB]      | Public  | Sub      | ResetStyle()  |
| [JScript] | public  | function | ResetStyle(); |

*Description*

**q) ToString**

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

*Description*

ListViewItem.ListViewSubItemCollection class (System.Windows.Forms)

**a) ToString**

*Description*

**b) *ListViewItem.ListViewSubItemCollection***

*Example Syntax:*

**c) *ToString***

```
[C#] public ListViewItem.ListViewSubItemCollection(ListViewItem owner);  
[C++] public: ListViewItemCollection(ListViewItem* owner);  
[VB] Public Sub New(ByVal owner As ListViewItem)  
[JScript] public function ListViewItem.ListViewSubItemCollection(owner :  
ListViewItem);
```

*Description*

**d) *Count***

**e) *ToString***

```
[C#] public int Count {get;}  
[C++] public: __property int get_Count();  
[VB] Public ReadOnly Property Count As Integer  
[JScript] public function get Count() : int;
```

*Description*

Returns the total number of items within the list view.  
*Return Value:* Integer count of items.

1        **f)      *IsReadOnly***

2        **g)      *ToString***

3  
4    [C#]        public        bool        IsReadOnly        {get;}

5    [C++]        public:        \_\_property        bool        get\_IsReadOnly();

6    [VB]    Public    ReadOnly    Property    IsReadOnly    As    Boolean

7    [JScript]    public    function    get    IsReadOnly()    :    Boolean;

8  
9    *Description*

10  
11        **h)      *Item***

12        **i)      *ToString***

13  
14    [C#]    public    ListViewItem.ListViewSubItem    this[int    index]    {get; set;}

15    [C++]    public:    \_\_property    ListViewItem.ListViewSubItem\*    get\_Item(int

16    index);public:        \_\_property        void        set\_Item(int        index,

17    ListViewItem.ListViewSubItem\*);

18    [VB]    Public    Default    Property    Item(ByVal    index    As    Integer)    As

19    ListViewItem.ListViewSubItem

20    [JScript]        returnValue        =

21    ListViewItemCollectionObject.Item(index);ListViewSubItemCollectionObject

22    .Item(index)        =        returnValue;

23  
24    *Description*

Returns a **System.Windows.Forms.ListView.ListViewSubItem** given it's zero based index into the **System.Windows.Forms.ListView.ListViewSubItemCollection** . The zero based index into list of sub items.

*j) Add*

[C#] public ListViewSubItem Add(ListViewItem.ListViewSubItem item);

[C++] public: ListViewSubItem\* Add(ListViewItem.ListViewSubItem\* item);

[VB] Public Function Add(ByVal item As ListViewItem.ListViewSubItem) As ListViewSubItem

[JScript] public function Add(item : ListViewItem.ListViewSubItem) : ListViewSubItem;

*Description*

*k) Add*

[C#] public ListViewSubItem Add(string text);

[C++] public: ListViewSubItem\* Add(String\* text);

[VB] Public Function Add(ByVal text As String) As ListViewSubItem

[JScript] public function Add(text : String) : ListViewSubItem;

*Description*

***l) Add***

```
[C#] public ListViewItem Add(string text, Color foreColor, Color backColor,
Font font);
[C++] public: ListViewItem* Add(String* text, Color foreColor, Color
BackColor, Font* font);
[VB] Public Function Add(ByVal text As String, ByVal foreColor As Color,
ByVal backColor As Color, ByVal font As Font) As ListViewItem
[JScript] public function Add(text : String, foreColor : Color, backColor : Color,
font : Font) : ListViewItem;
```

***Description***

***m) AddRange***

```
[C#] public void AddRange(ListViewItem.ListViewSubItem[] items);
[C++] public: void AddRange(ListViewItem.ListViewSubItem* items[]);
[VB] Public Sub AddRange(ByVal items() As ListViewItem.ListViewSubItem)
[JScript] public function AddRange(items : ListViewItem.ListViewSubItem[]);
```

***Description***

***n) AddRange***

```
[C#] public void AddRange(string[] items);
```

```

1 [C++]      public:      void      AddRange(String*      items      __gc[]);
2 [VB]      Public      Sub      AddRange(ByVal      items()      As      String)
3 [JScript]      public      function      AddRange(items      :      String[]);

```

#### 5 *Description*

##### 7 *o) AddRange*

```

9 [C#] public void AddRange(string[] items, Color foreColor, Color backColor,
10 Font font);
11 [C++] public: void AddRange(String* items __gc[], Color foreColor, Color
12 backColor, Font* font);
13 [VB] Public Sub AddRange(ByVal items() As String, ByVal foreColor As Color,
14 ByVal backColor As Color, ByVal font As Font)
15 [JScript] public function AddRange(items : String[], foreColor : Color, backColor
16 : Color, font : Font);

```

#### 18 *Description*

##### 20 *p) Clear*

```

22 [C#]      public      void      Clear();
23 [C++]      public:      __sealed      void      Clear();
24 [VB]      NotOverridable      Public      Sub      Clear()
25 [JScript]      public      function      Clear();

```

1  
2 *Description*

3  
4 *q) Contains*

5  
6 [C#] public bool Contains(ListViewItem.ListViewSubItem subItem);  
7 [C++] public: bool Contains(ListViewItem.ListViewSubItem\* subItem);  
8 [VB] Public Function Contains(ByVal subItem As  
9 ListViewItem.ListViewSubItem) As Boolean  
10 [JScript] public function Contains(subItem : ListViewItem.ListViewSubItem) :  
11 Boolean;  
12

13 *Description*

14  
15 *r) GetEnumerator*

16  
17 [C#] public IEnumerator GetEnumerator();  
18 [C++] public: \_\_sealed IEnumerator\* GetEnumerator();  
19 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator  
20 [JScript] public function GetEnumerator() : IEnumerator;  
21

22 *Description*  
23  
24  
25



s) *IndexOf*

```
[C#]    public    int    IndexOf(ListViewItem.ListViewSubItem    subItem);  
[C++]   public:   int    IndexOf(ListViewItem.ListViewSubItem*    subItem);  
[VB]    Public    Function    IndexOf(ByVal    subItem    As  
ListViewItem.ListViewSubItem)                As                Integer  
[JScript] public function IndexOf(subItem : ListViewItem.ListViewSubItem) : int;
```

*Description*

t) *Insert*

```
[C#]    public    void    Insert(int    index, ListViewItem.ListViewSubItem    item);  
[C++]   public:   void    Insert(int    index, ListViewItem.ListViewSubItem*    item);  
[VB]    Public    Sub    Insert(ByVal    index    As    Integer, ByVal    item    As  
ListViewItem.ListViewSubItem)  
[JScript]    public    function    Insert(index    :    int,    item    :  
ListViewItem.ListViewSubItem);
```

*Description*

u) *Remove*

```
[C#]    public    void    Remove(ListViewItem.ListViewSubItem    item);  
[C++]   public:   void    Remove(ListViewItem.ListViewSubItem*    item);
```

1 [VB] Public Sub Remove(ByVal item As ListViewItem.ListViewSubItem)

2 [JScript] public function Remove(item : ListViewItem.ListViewSubItem);

3  
4 *Description*

5  
6 **v) RemoveAt**

7  
8 [C#] public void RemoveAt(int index);

9 [C++] public: \_\_sealed void RemoveAt(int index);

10 [VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

11 [JScript] public function RemoveAt(index : int);

12  
13 *Description*

14  
15 **w) ICollection.CopyTo**

16  
17 [C#] void ICollection.CopyTo(Array dest, int index);

18 [C++] void ICollection::CopyTo(Array\* dest, int index);

19 [VB] Sub CopyTo(ByVal dest As Array, ByVal index As Integer) Implements  
20 ICollection.CopyTo

21 [JScript] function ICollection.CopyTo(dest : Array, index : int);

22 **x) IList.Add**

23  
24 [C#] int IList.Add(object item);

```

1  [C++]          int          IList::Add(Object*          item);
2  [VB] Function Add(ByVal item As Object) As Integer Implements IList.Add
3  [JScript] function IList.Add(item : Object) : int;
4
5
6  y)  IList.Contains
7
8  [C#]          bool          IList.Contains(object          subItem);
9  [C++]          bool          IList::Contains(Object*          subItem);
10 [VB] Function Contains(ByVal subItem As Object) As Boolean Implements
11      IList.Contains
12 [JScript] function IList.Contains(subItem : Object) : Boolean;
13
14 z)  IList.IndexOf
15
16 [C#]          int          IList.IndexOf(object          subItem);
17 [C++]          int          IList::IndexOf(Object*          subItem);
18 [VB] Function IndexOf(ByVal subItem As Object) As Integer Implements
19      IList.IndexOf
20 [JScript] function IList.IndexOf(subItem : Object) : int;
21
22 aa)  IList.Insert
23
24 [C#]          void          IList.Insert(int          index,          object          item);
25 [C++]          void          IList::Insert(int          index,          Object*          item);
26 [VB] Sub Insert(ByVal index As Integer, ByVal item As Object) Implements
27      IList.Insert
28 [JScript] function IList.Insert(index : int, item : Object);

```

*bb) IList.Remove*

```
1
2 [C#]          void          IList.Remove(object          item);
3
4 [C++]          void          IList::Remove(Object*          item);
5
6 [VB] Sub Remove(ByVal item As Object) Implements IList.Remove
7
8 [JScript] function IList.Remove(item : Object);
9
10 MainMenu class (System.Windows.Forms)
```

*a) ToString*

*Description*

Represents the menu structure of a form.

The **System.Windows.Forms.MainMenu** control represents the container for the menu structure of a form. A menu is composed of **System.Windows.Forms.MenuItem** objects that represent the individual menu commands in the menu structure. Each **System.Windows.Forms.MenuItem** can be a command for your application or a parent menu for other submenu items. To bind the **System.Windows.Forms.MainMenu** to the **System.Windows.Forms.Form** that will display it, assign the **System.Windows.Forms.MainMenu** to the **System.Windows.Forms.Form.Menu** property of the **System.Windows.Forms.Form**.

*b) MainMenu*

*Example Syntax:*

*c) ToString*

```
22
23 [C#]          public          MainMenu();
24
25 [C++]          public:          MainMenu();
26
27 [VB]          Public          Sub          New()
```

[JScript] public function MainMenu(); Initializes a new instance of the **System.Windows.Forms.MainMenu** class.

#### *Description*

Initializes a new instance of the **System.Windows.Forms.MainMenu** class without any specified menu items.

This version of the constructor creates a **System.Windows.Forms.MainMenu** without any specified **System.Windows.Forms.MenuItem** objects. To add menu items to the control use the other version of this constructor that accepts an array of **System.Windows.Forms.MenuItem** objects as its parameter or use the

**System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)** method of the **System.Windows.Forms.Menu.MenuItems** property.

#### *d) MainMenu*

*Example Syntax:*

#### *e) ToString*

|           |                      |                     |                |
|-----------|----------------------|---------------------|----------------|
| [C#]      | public               | MainMenu(MenuItem[] | items);        |
| [C++]     | public:              | MainMenu(MenuItem*  | items[]);      |
| [VB]      | Public Sub New(ByVal | items() As          | MenuItem)      |
| [JScript] | public function      | MainMenu(items      | : MenuItem[]); |

#### *Description*

Initializes a new instance of the **System.Windows.Forms.MainMenu** with a specified set of **System.Windows.Forms.MenuItem** objects.

You can use this constructor to assign an array of **System.Windows.Forms.MenuItem** objects to the **System.Windows.Forms.MainMenu** at the time of its creation. After the **System.Windows.Forms.MainMenu** has been created you can add additional **System.Windows.Forms.MenuItem** objects to the **System.Windows.Forms.MainMenu** using the

**System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)**  
 method of the **System.Windows.Forms.Menu.MenuItems** property. An  
 array of **System.Windows.Forms.MenuItem** objects that will be added to the  
**System.Windows.Forms.MainMenu** .

- f) Container*
- g) DesignMode*
- h) Events*
- i) Handle*
- j) IsParent*
- k) MdiListItem*
- l) MenuItems*
- m) RightToLeft*
- n) ToString*

#### *Description*

Gets or sets whether the text displayed by the control is displayed from right to left.

This property allows your menus to support languages that are written from right to left. When this property is set to **RightToLeft.Yes** , the menu item text will be displayed from right to left instead of the default left to right method.

- o) Site*
- p) CloneMenu*

|       |             |         |           |                         |
|-------|-------------|---------|-----------|-------------------------|
| [C#]  | public      | virtual | MainMenu  | CloneMenu();            |
| [C++] | public:     | virtual | MainMenu* | CloneMenu();            |
| [VB]  | Overridable | Public  | Function  | CloneMenu() As MainMenu |

1 [JScript] public function CloneMenu() : MainMenu;

3 *Description*

4 Creates a new **System.Windows.Forms.MainMenu** that is a duplicate of the  
current **System.Windows.Forms.MainMenu** .

5 *Return Value:* A **System.Windows.Forms.MainMenu** that represents the  
cloned menu.

6 You can use this method to create a copy of the menu structure stored in a  
7 **System.Windows.Forms.MainMenu** . You can use this method to reuse the  
menu structure stored in a **System.Windows.Forms.MainMenu** as the  
8 foundation for a new **System.Windows.Forms.MainMenu** . For example, if  
you want to create a menu structure that has the same menu items as an  
9 existing **System.Windows.Forms.MainMenu** but will also have additional  
10 **System.Windows.Forms.MenuItem** objects added to it, you can use the  
**System.Windows.Forms.MainMenu.CloneMenu** method to create a copy of  
11 the original **System.Windows.Forms.MainMenu** and then add the new  
**System.Windows.Forms.MenuItem** objects to the cloned  
12 **System.Windows.Forms.MainMenu** .

13 *q) CreateMenuHandle*

15 [C#] protected override IntPtr CreateMenuHandle();

16 [C++] protected: IntPtr CreateMenuHandle();

17 [VB] Overrides Protected Function CreateMenuHandle() As IntPtr

18 [JScript] protected override function CreateMenuHandle() : IntPtr;

20 *Description*

21 *r) Dispose*

23 [C#] protected override void Dispose(bool disposing);

24 [C++] protected: void Dispose(bool disposing);

[VB] Overrides Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected override function Dispose(disposing : Boolean);

#### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.MainMenu**.

Call **System.Windows.Forms.MainMenu.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.MainMenu**. The **System.Windows.Forms.MainMenu.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.MainMenu** in an unusable state. After calling **System.Windows.Forms.MainMenu.Dispose(System.Boolean)**, you must release all references to the **System.Windows.Forms.MainMenu** so the memory it was occupying can be reclaimed by garbage collection.

#### *s) GetForm*

[C#] public Form GetForm();

[C++] public: Form\* GetForm();

[VB] Public Function GetForm() As Form

[JScript] public function GetForm() : Form;

#### *Description*

Gets the **System.Windows.Forms.Form** that contains this control.

*Return Value:* A **System.Windows.Forms.Form** that is the container for this control. Returns **null** if the **System.Windows.Forms.MainMenu** is not currently hosted on a form.

This property enables you to determine if a specific **System.Windows.Forms.MainMenu** is parented to a form. The property is typically used when multiple **System.Windows.Forms.MainMenu** objects are being used on a form and you need to determine which one is currently being used by a form.



#### t) *ToString*

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

#### *Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

MdiClient class (System.Windows.Forms)

#### a) *ToString*

#### *Description*

Summary to Come

The MdiClient is that window that contains MDI children. This control should not be created directly, but rather only used by System.Windows.Forms.Form.

#### b) *MdiClient*

*Example Syntax:*

#### c) *ToString*

|           |         |                       |
|-----------|---------|-----------------------|
| [C#]      | public  | MdiClient();          |
| [C++]     | public: | MdiClient();          |
| [VB]      | Public  | Sub New()             |
| [JScript] | public  | function MdiClient(); |

*Description*

Creates a new MdiClient.

- d) *AccessibilityObject*
- e) *AccessibleDefault.ActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *BackColor*
- l) *BackgroundImage*
- m) *ToString*

*Description*

Use parent's BackgroundImage if our BackgroundImage isn't set.

- n) *BindingContext*
- o) *Bottom*
- p) *Bounds*
- q) *CanFocus*
- r) *CanSelect*
- s) *Capture*
- t) *CausesValidation*
- u) *ClientRectangle*
- v) *ClientSize*
- w) *CompanyName*
- x) *Container*
- y) *ContainsFocus*
- z) *ContextMenu*
- aa) *Controls*
- bb) *Created*
- cc) *CreateParams*
- dd) *ToString*

*Description*

**ee) Cursor**  
**ff) DataBindings**  
**gg) DefaultImeMode**  
**hh) DefaultSize**  
**ii) DesignMode**  
**jj) DisplayRectangle**  
**kk) Disposing**  
**ll) Dock**  
**mm) Enabled**  
**nn) Events**  
**oo) Focused**  
**pp) Font**  
**qq) FontHeight**  
**rr) ForeColor**  
**ss) Handle**  
**tt) HasChildren**  
**uu) Height**  
**vv) ImeMode**  
**ww) InvokeRequired**  
**xx) IsAccessible**  
**yy) IsDisposed**  
**zz) IsHandleCreated**  
**aaa) Left**

*bbb) Location*

*ccc) MdiChildren*

*ddd) ToString*

*Description*

The list of MDI children contained. This list will be sorted by the order in which the children were added to the form, not the current ZOrder.

|    |                               |
|----|-------------------------------|
| 1  | <i>eee) Name</i>              |
| 2  | <i>fff) Parent</i>            |
| 3  | <i>ggg) ProductName</i>       |
| 4  | <i>hhh) ProductVersion</i>    |
| 5  | <i>iii) RecreatingHandle</i>  |
| 6  | <i>jjj) Region</i>            |
| 7  | <i>kkk) RenderRightToLeft</i> |
| 8  | <i>lll) ResizeRedraw</i>      |
| 9  | <i>mmm) Right</i>             |
| 10 | <i>nnn) RightToLeft</i>       |
| 11 | <i>ooo) ShowFocusCues</i>     |
| 12 | <i>ppp) ShowKeyboardCues</i>  |
| 13 | <i>qqq) Site</i>              |
| 14 | <i>rrr) Size</i>              |
| 15 | <i>sss) TabIndex</i>          |
| 16 | <i>ttt) TabStop</i>           |
| 17 | <i>uuu) Tag</i>               |
| 18 | <i>vvv) Text</i>              |
| 19 | <i>www) Top</i>               |
| 20 | <i>xxx) TopLevelControl</i>   |
| 21 | <i>yyy) Visible</i>           |
| 22 | <i>zzz) Width</i>             |
| 23 | <i>aaaa) WindowTarget</i>     |
| 24 |                               |
| 25 |                               |

**bbbb) CreateControlsInstance**

```
[C#]    protected    override    ControlCollection    CreateControlsInstance();
[C++]    protected:    ControlCollection*    CreateControlsInstance();
[VB] Overrides Protected Function CreateControlsInstance() As ControlCollection
[JScript] protected override function CreateControlsInstance() : ControlCollection;
```

*Description*

**cccc) LayoutMdi**

```
[C#]    public    void    LayoutMdi(MdiLayout    value);
[C++]    public:    void    LayoutMdi(MdiLayout    value);
[VB]    Public    Sub    LayoutMdi(ByVal    value    As    MdiLayout)
[JScript]    public    function    LayoutMdi(value    :    MdiLayout);
```

*Description*

Arranges the MDI child forms according to value, which should be a member of the MdiLayout enum. Any value from the MdiLayout enum class.

**dddd) OnResize**

```
[C#]    protected    override    void    OnResize(EventArgs    e);
[C++]    protected:    void    OnResize(EventArgs*    e);
[VB] Overrides Protected Sub OnResize(ByVal e As EventArgs)
[JScript]    protected    override    function    OnResize(e    :    EventArgs);
```

1  
2 *Description*

3  
4 *eeee) ScaleCore*

5  
6 [C#] protected override void ScaleCore(float dx, float dy);

7 [C++] protected: void ScaleCore(float dx, float dy);

8 [VB] Overrides Protected Sub ScaleCore(ByVal dx As Single, ByVal dy As  
9 Single)

10 [JScript] protected override function ScaleCore(dx : float, dy : float);

11  
12 *Description*

13 Performs the work of scaling the entire control and any child controls. Ratio to  
14 scale the control horizontally. Ratio to scale the control vertically.

15 *ffff) SetBoundsCore*

16 [C#] protected override void SetBoundsCore(int x, int y, int width, int height,  
17 BoundsSpecified specified);

18 [C++] protected: void SetBoundsCore(int x, int y, int width, int height,  
19 BoundsSpecified specified);

20 [VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As  
21 Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As  
22 BoundsSpecified)

23 [JScript] protected override function SetBoundsCore(x : int, y : int, width : int,  
24 height : int, specified : BoundsSpecified);



1  
2 *Description*

3  
4 *gggg) WndProc*

5  
6 [C#]      protected      override      void      WndProc(ref      Message      m);

7 [C++]           protected:           void           WndProc(Message\*      m);

8 [VB]      Overrides      Protected      Sub      WndProc(ByRef      m      As      Message)

9 [JScript]      protected      override      function      WndProc(m      :      Message);

10  
11 *Description*

12  
13      MdiLayout enumeration (System.Windows.Forms)

14 *a)      WndProc*

15  
16  
17 *Description*

18 Specifies the layout of multiple document interface (MDI) child windows in an MDI parent window.

19 Use the members of this enumeration when calling the  
20 **System.Windows.Forms.Form.LayoutMdi(System.Windows.Forms.MdiLayout)** method of the **System.Windows.Forms.Form** class.

21  
22 *b)      WndProc*

23  
24 [C#]           public           const           MdiLayout           ArrangeIcons;

25 [C++]           public:           const           MdiLayout           ArrangeIcons;

|           |        |       |              |    |            |
|-----------|--------|-------|--------------|----|------------|
| [VB]      | Public | Const | ArrangeIcons | As | MdiLayout  |
| [JScript] | public | var   | Arrangelcons | :  | MdiLayout; |

*Description*

All MDI child icons are arranged within the client region of the MDI parent form.

*c) WndProc*

|           |         |       |           |    |            |
|-----------|---------|-------|-----------|----|------------|
| [C#]      | public  | const | MdiLayout |    | Cascade;   |
| [C++]     | public: | const | MdiLayout |    | Cascade;   |
| [VB]      | Public  | Const | Cascade   | As | MdiLayout  |
| [JScript] | public  | var   | Cascade   | :  | MdiLayout; |

*Description*

All MDI child windows are cascaded within the client region of the MDI parent form.

*d) WndProc*

|           |         |       |                |    |                 |
|-----------|---------|-------|----------------|----|-----------------|
| [C#]      | public  | const | MdiLayout      |    | TileHorizontal; |
| [C++]     | public: | const | MdiLayout      |    | TileHorizontal; |
| [VB]      | Public  | Const | TileHorizontal | As | MdiLayout       |
| [JScript] | public  | var   | TileHorizontal | :  | MdiLayout;      |

*Description*

All MDI child windows are tiled horizontally within the client region of the MDI parent form.

e) *WndProc*

|           |         |       |              |               |
|-----------|---------|-------|--------------|---------------|
| [C#]      | public  | const | MdiLayout    | TileVertical; |
| [C++]     | public: | const | MdiLayout    | TileVertical; |
| [VB]      | Public  | Const | TileVertical | As MdiLayout  |
| [JScript] | public  | var   | TileVertical | : MdiLayout;  |

*Description*

All MDI child windows are tiled vertically within the client region of the MDI parent form.

MeasureItemEventArgs class (System.Windows.Forms)

a) *ToString*

*Description*

Provides data for the **MeasureItem** event of the **System.Windows.Forms.ListBox** , **System.Windows.Forms.ComboBox** , **System.Windows.Forms.CheckedListBox** , and **System.Windows.Forms.MenuItem** controls.

This event is sent when the **OwnerDraw** property of **System.Windows.Forms.ListBox** , **System.Windows.Forms.ComboBox** , **System.Windows.Forms.CheckedListBox** , or **System.Windows.Forms.MenuItem** is set to **true** . It is used to tell the drawing function how to size an item.

b) *MeasureItemEventArgs*

*Example Syntax:*

c) *ToString*

```
[C#] public MeasureItemEventArgs(Graphics graphics, int index);  
[C++] public: MeasureItemEventArgs(Graphics* graphics, int index);  
[VB] Public Sub New(ByVal graphics As Graphics, ByVal index As Integer)  
[JScript] public function MeasureItemEventArgs(graphics : Graphics, index : int);
```

*Description*

Initializes a new instance of the **System.Windows.Forms.MeasureItemEventArgs** class. The **System.Drawing.Graphics** object being written to. The index of the item for which you need the height or width.

d) *MeasureItemEventArgs*

*Example Syntax:*

e) *ToString*

```
[C#] public MeasureItemEventArgs(Graphics graphics, int index, int itemHeight);  
[C++] public: MeasureItemEventArgs(Graphics* graphics, int index, int  
itemHeight);  
[VB] Public Sub New(ByVal graphics As Graphics, ByVal index As Integer,  
ByVal itemHeight As Integer)  
[JScript] public function MeasureItemEventArgs(graphics : Graphics, index : int,  
itemHeight : int); Initializes a new instance of the  
System.Windows.Forms.MeasureItemEventArgs class.
```

*Description*

Initializes a new instance of the

**System.Windows.Forms.MeasureItemEventArgs** class providing a parameter for the item height. The **System.Drawing.Graphics** object being written to. The index of the item for which you need the height or width. The height of the item to measure relative to the *graphics* object.

f) *Graphics*

g) *ToString*

[C#]            public            Graphics            Graphics            {get;}

[C++]           public:            \_\_property           Graphics\*           get\_Graphics();

[VB]       Public       ReadOnly       Property       Graphics       As       Graphics

[JScript]       public       function       get       Graphics()       :       Graphics;

### *Description*

Gets the **System.Drawing.Graphics** object to measure against.

You use a **System.Drawing.Graphics** object to determine the scale to use when setting the

**System.Windows.Forms.MeasureItemEventArgs.ItemHeight** and **System.Windows.Forms.MeasureItemEventArgs.ItemWidth** . Different types of graphics objects can have different scales, such as the difference in measurement scale between a monitor screen and a printer.

h) *Index*

i) *ToString*

[C#]            public            int            Index            {get;}

[C++]           public:            \_\_property           int            get\_Index();

[VB]       Public       ReadOnly       Property       Index       As       Integer

[JScript]       public       function       get       Index()       :       int;

1  
2 *Description*

3 Gets or sets the index of the item for which the height and width is needed.

4       j)       *ItemHeight*

5       k)       *ToString*

6  
7 [C#]       public       int       ItemHeight       {get;       set;}

8 [C++] public: \_\_property int get\_ItemHeight();public: \_\_property void  
9 set\_ItemHeight(int);

10 [VB]       Public       Property       ItemHeight       As       Integer

11 [JScript] public function get ItemHeight() : int;public function set ItemHeight(int);

12  
13 *Description*

14 Gets or sets the height of the item specified by the  
15 **System.Windows.Forms.MeasureItemEventArgs.Index** .

16       l)       *ItemWidth*

17       m)       *ToString*

18  
19 [C#]       public       int       ItemWidth       {get;       set;}

20 [C++] public: \_\_property int get\_ItemWidth();public: \_\_property void  
21 set\_ItemWidth(int);

22 [VB]       Public       Property       ItemWidth       As       Integer

23 [JScript] public function get ItemWidth() : int;public function set ItemWidth(int);

24  
25 *Description*

Gets or sets the width of the item specified by the  
**System.Windows.Forms.MeasureItemEventArgs.Index** .

This member is only used by **System.Windows.Forms.MenuItem** . You use this property to ensure the menu is at least as wide as the widest menu item in the menu list.

MeasureItemEventHandler delegate (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the method that will handle the **MeasureItem** event of the **System.Windows.Forms.ListBox** , **System.Windows.Forms.ComboBox** , **System.Windows.Forms.CheckedListBox** , or **System.Windows.Forms.MenuItem** controls. The source of the event. A **System.Windows.Forms.MeasureItemEventArgs** that contains the event data.

When you create a **System.Windows.Forms.MeasureItemEventHandler** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

Menu class (System.Windows.Forms)

a) *ToString*

#### *Description*

Represents the base functionality for all menus.

This class is the base class for the **System.Windows.Forms.MainMenu** , **System.Windows.Forms.MenuItem** , and **System.Windows.Forms.ContextMenu** classes. You cannot create an instance of this class. The menus for an application are comprised of **System.Windows.Forms.MenuItem** objects. These can contain other **System.Windows.Forms.MenuItem** objects, representing submenu items.

The **System.Windows.Forms.MenuItem** objects can be stored in a **System.Windows.Forms.MainMenu** for display as an entire menu structure for a form or a **System.Windows.Forms.ContextMenu** that is used to display shortcut menus. This class provides functionality that is common for all the menu classes.

*b) ToString*

|           |         |       |            |             |
|-----------|---------|-------|------------|-------------|
| [C#]      | public  | const | int        | FindHandle; |
| [C++]     | public: | const | int        | FindHandle; |
| [VB]      | Public  | Const | FindHandle | As Integer  |
| [JScript] | public  | var   | FindHandle | : int;      |

*Description*

Used by findMenuItem Used by findMenuItem

*c) ToString*

|           |         |       |              |               |
|-----------|---------|-------|--------------|---------------|
| [C#]      | public  | const | int          | FindShortcut; |
| [C++]     | public: | const | int          | FindShortcut; |
| [VB]      | Public  | Const | FindShortcut | As Integer    |
| [JScript] | public  | var   | FindShortcut | : int;        |

*Description*

Used by findMenuItem Used by findMenuItem

*d) Menu*

*Example Syntax:*



e) *ToString*

```
[C#]          protected          Menu(MenuItem[]          items);
[C++]          protected:          Menu(MenuItem*          items[]);
[VB] Protected Sub New(ByVal items() As MenuItem)
[JScript] protected function Menu(items : MenuItem[]);
```

*Description*

This is an abstract class. Instances cannot be created, so the constructor is only called from derived classes.

f) *Container*

g) *DesignMode*

h) *Events*

i) *Handle*

j) *ToString*

*Description*

Gets a value representing the window handle for the menu.

You can use this property to obtain the handle to the menu to perform special operations to the menu outside of the functionality provided by this class or its derived classes.

k) *IsParent*

l) *ToString*

```
[C#]          public          virtual          bool          IsParent          {get;}
```

```

1 [C++]      public:      __property      virtual      bool      get_IsParent();
2 [VB]  Overridable  Public  ReadOnly  Property  IsParent  As  Boolean
3 [JScript]      public      function      get      IsParent()      :      Boolean;

```

#### *Description*

Gets a value indicating whether this menu contains any menu items. This property is read-only.

You can use this method to determine whehter there are any **System.Windows.Forms.MenuItem** objects assigned to this menu. This is equivalent to checking for **null** in the **System.Windows.Forms.Menu.MenuItems** property.

*m) MdiListItem*

*n) ToString*

```

13 [C#]      public      MenuItem      MdiListItem      {get;}
14 [C++]      public:      __property      MenuItem*      get_MdiListItem();
15 [VB]  Public  ReadOnly  Property  MdiListItem  As  MenuItem
16 [JScript]      public      function      get      MdiListItem()      :      MenuItem;

```

#### *Description*

Gets a value indicating the **System.Windows.Forms.MenuItem** that is used to display a list of Multiple Document Interface (MDI) child forms.

You can use this property to determine whether a **System.Windows.Forms.MenuItem** has been specified to display the list of open child windows in an MDI application. To use a specific **System.Windows.Forms.MenuItem** as an MDI list, set the **System.Windows.Forms.MenuItem.MdiList** property in the **System.Windows.Forms.MenuItem** to be used.

o) *MenuItems*

p) *ToString*

```
[C#]      public      Menu.MenuItemCollection      MenuItems      {get;}
[C++]     public:  __property  Menu.MenuItemCollection*  get_MenuItems();
[VB]      Public ReadOnly Property MenuItems As Menu.MenuItemCollection
[JScript] public  function  get  MenuItems() : Menu.MenuItemCollection;
```

#### *Description*

Gets a value indicating the collection of **System.Windows.Forms.MenuItem** objects associated with the menu.

You can use this property to obtain a reference to the list of menu items that are currently stored in the menu. For **System.Windows.Forms.MainMenu** and **System.Windows.Forms.ContextMenu** objects, the **System.Windows.Forms.Menu.MenuItems** property contains the entire menu structure in the control. For the **System.Windows.Forms.MenuItem** class, the **System.Windows.Forms.Menu.MenuItems** property contains the list of submenu items associated with the **System.Windows.Forms.MenuItem**. With the reference to the collection of menu items for the menu (provided by this property), you can add and remove menu items, determine the total number of menu items, and clear the list of menu items from the collection. For more information on maintaining the menu item collection for a menu, see the **System.Windows.Forms.Menu.MenuItemCollection** documentation.

q) *Site*

r) *CloneMenu*

```
[C#]      protected      void      CloneMenu(Menu      menuSrc);
[C++]     protected:      void      CloneMenu(Menu*      menuSrc);
[VB]      Protected      Sub      CloneMenu(ByVal      menuSrc      As      Menu)
[JScript] protected      function      CloneMenu(menuSrc      :      Menu);
```

## Description

Copies the **System.Windows.Forms.Menu** that is passed as a parameter to the current **System.Windows.Forms.Menu** .

This method copies the entire list of **System.Windows.Forms.MenuItem** objects (stored in the **System.Windows.Forms.Menu** passed in to *menuSrc* ) into the current menu. You can use this method in your derived class to clone **System.Windows.Forms.MenuItem** objects. They can then be reused by other classes that derive from **System.Windows.Forms.Menu** , such as **System.Windows.Forms.MainMenu** , **System.Windows.Forms.ContextMenu** , and **System.Windows.Forms.MenuItem** . The **System.Windows.Forms.Menu** to copy.

### s) CreateMenuHandle

|           |             |           |                    |                              |
|-----------|-------------|-----------|--------------------|------------------------------|
| [C#]      | protected   | virtual   | IntPtr             | CreateMenuHandle();          |
| [C++]     | protected:  | virtual   | IntPtr             | CreateMenuHandle();          |
| [VB]      | Overridable | Protected | Function           | CreateMenuHandle() As IntPtr |
| [JScript] | protected   | function  | CreateMenuHandle() | : IntPtr;                    |

## Description

### t) Dispose

|           |            |           |                          |                                     |
|-----------|------------|-----------|--------------------------|-------------------------------------|
| [C#]      | protected  | override  | void                     | Dispose(bool disposing);            |
| [C++]     | protected: | void      | Dispose(bool disposing); |                                     |
| [VB]      | Overrides  | Protected | Sub                      | Dispose(ByVal disposing As Boolean) |
| [JScript] | protected  | override  | function                 | Dispose(disposing : Boolean);       |

## Description

Disposes of the resources (other than memory) used by the  
**System.Windows.Forms.Menu** .

Call **System.Windows.Forms.Menu.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.Menu** . The **System.Windows.Forms.Menu.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.Menu** in an unusable state. After calling **System.Windows.Forms.Menu.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.Menu** so the memory it was occupying can be reclaimed by garbage collection.

*u) FindMenuItem*

[C#] public MenuItem FindMenuItem(int type, IntPtr value);

[C++] public: MenuItem\* FindMenuItem(int type, IntPtr value);

[VB] Public Function FindMenuItem(ByVal type As Integer, ByVal value As IntPtr) As MenuItem

[JScript] public function FindMenuItem(type : int, value : IntPtr) : MenuItem;

*Description*

*v) FindMergePosition*

[C#] protected int FindMergePosition(int mergeOrder);

[C++] protected: int FindMergePosition(int mergeOrder);

[VB] Protected Function FindMergePosition(ByVal mergeOrder As Integer) As Integer

[JScript] protected function FindMergePosition(mergeOrder : int) : int;

*Description*

### w) *GetContextMenu*

|           |         |              |                                 |
|-----------|---------|--------------|---------------------------------|
| [C#]      | public  | ContextMenu  | GetContextMenu();               |
| [C++]     | public: | ContextMenu* | GetContextMenu();               |
| [VB]      | Public  | Function     | GetContextMenu() As ContextMenu |
| [JScript] | public  | function     | GetContextMenu() : ContextMenu; |

#### *Description*

Gets the **System.Windows.Forms.ContextMenu** that contains this menu.  
*Return Value:* The **System.Windows.Forms.ContextMenu** that contains this menu. The default is **null**.

This method allows you to obtain a reference to the **System.Windows.Forms.ContextMenu** that this menu is contained in. This property returns **null** if the menu is not contained in a **System.Windows.Forms.ContextMenu**. This can occur if the menu is contained in a **System.Windows.Forms.MenuItem** or **System.Windows.Forms.MainMenu**, or if the menu is not contained in any menu. You can use this property to determine whether a menu is currently being used, and also to determine where.

### x) *GetMainMenu*

|           |         |           |                           |
|-----------|---------|-----------|---------------------------|
| [C#]      | public  | MainMenu  | GetMainMenu();            |
| [C++]     | public: | MainMenu* | GetMainMenu();            |
| [VB]      | Public  | Function  | GetMainMenu() As MainMenu |
| [JScript] | public  | function  | GetMainMenu() : MainMenu; |

#### *Description*

Gets the **System.Windows.Forms.MainMenu** that contains this menu.  
*Return Value:* The **System.Windows.Forms.MainMenu** that contains this menu.

This method allows you to obtain a reference to the **System.Windows.Forms.MainMenu** this menu is currently located in. This property returns **null** if the menu is not contained in a **System.Windows.Forms.MainMenu**. This can occur if the menu is contained in a **System.Windows.Forms.MenuItem** or **System.Windows.Forms.ContextMenu**, or if the menu is not contained in any menu. You can use this property to determine whether a menu is currently being used, and also to determine where.

#### y) *MergeMenu*

```
[C#]      public      virtual      void      MergeMenu(Menu      menuSrc);
[C++]     public:     virtual      void      MergeMenu(Menu*      menuSrc);
[VB]      Overridable Public Sub MergeMenu(ByVal menuSrc As Menu)
[JScript] public      function      MergeMenu(menuSrc      :      Menu);
```

#### *Description*

Merges the **System.Windows.Forms.MenuItem** objects of one menu with the current menu.

This method merges **System.Windows.Forms.MenuItem** objects from one menu with the current menu. The **System.Windows.Forms.Menu** whose menu items are merged with the menu items of the current menu.

#### z) *ProcessCmdKey*

```
[C#] protected virtual bool ProcessCmdKey(ref Message msg, Keys keyData);
[C++] protected: virtual bool ProcessCmdKey(Message* msg, Keys keyData);
[VB] Overridable Protected Function ProcessCmdKey(ByRef msg As Message,
ByVal      keyData      As      Keys)      As      Boolean
[JScript] protected function ProcessCmdKey(msg : Message, keyData : Keys) :
Boolean;
```

*Description*

*aa) ToString*

|           |           |          |          |                      |
|-----------|-----------|----------|----------|----------------------|
| [C#]      | public    | override | string   | ToString();          |
| [C++]     | public:   |          | String*  | ToString();          |
| [VB]      | Overrides | Public   | Function | ToString() As String |
| [JScript] | public    | override | function | ToString() : String; |

*Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

MenuGlyph enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the image to draw when drawing a menu with the **System.Windows.Forms.ControlPaint.DrawMenuGlyph(System.Drawing.Graphics,System.Drawing.Rectangle,System.Windows.Forms.MenuGlyph)** method.

The values of this enumeration are used in the **System.Windows.Forms.ControlPaint.DrawMenuGlyph(System.Drawing.Graphics,System.Drawing.Rectangle,System.Windows.Forms.MenuGlyph)** method of the **System.Windows.Forms.ControlPaint** class. These values represent the different types of symbols that can be drawn on a menu item.



**b) ToString**

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuGlyph | Arrow;       |
| [C++]     | public: | const | MenuGlyph | Arrow;       |
| [VB]      | Public  | Const | Arrow     | As MenuGlyph |
| [JScript] | public  | var   | Arrow     | : MenuGlyph; |

*Description*

Draws a submenu arrow.

**c) ToString**

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuGlyph | Bullet;      |
| [C++]     | public: | const | MenuGlyph | Bullet;      |
| [VB]      | Public  | Const | Bullet    | As MenuGlyph |
| [JScript] | public  | var   | Bullet    | : MenuGlyph; |

*Description*

Draws a menu bullet.

**d) ToString**

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuGlyph | Checkmark;   |
| [C++]     | public: | const | MenuGlyph | Checkmark;   |
| [VB]      | Public  | Const | Checkmark | As MenuGlyph |
| [JScript] | public  | var   | Checkmark | : MenuGlyph; |

*Description*

Draws a menu check mark.

*e) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuGlyph | Max;         |
| [C++]     | public: | const | MenuGlyph | Max;         |
| [VB]      | Public  | Const | Max       | As MenuGlyph |
| [JScript] | public  | var   | Max       | : MenuGlyph; |

*Description*

*f) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuGlyph | Min;         |
| [C++]     | public: | const | MenuGlyph | Min;         |
| [VB]      | Public  | Const | Min       | As MenuGlyph |
| [JScript] | public  | var   | Min       | : MenuGlyph; |

*Description*

MenuItem class (System.Windows.Forms)

a) *ToString*

*Description*

Represents an individual item that is displayed within a **System.Windows.Forms.MainMenu** or **System.Windows.Forms.ContextMenu** .

In order for a **System.Windows.Forms.MenuItem** to be displayed, you must add it to a **System.Windows.Forms.MainMenu** or **System.Windows.Forms.ContextMenu** . To create submenus, you can add **System.Windows.Forms.MenuItem** objects to the **System.Windows.Forms.Menu.MenuItems** property of the parent **System.Windows.Forms.MenuItem** .

b) *MenuItem*

*Example Syntax:*

c) *ToString*

|           |                 |             |
|-----------|-----------------|-------------|
| [C#]      | public          | MenuItem(); |
| [C++]     | public:         | MenuItem(); |
| [VB]      | Public          | Sub New()   |
| [JScript] | public function | MenuItem(); |

Initializes a new instance of the **System.Windows.Forms.MenuItem** class.

*Description*

Initializes a **System.Windows.Forms.MenuItem** with a blank caption.

Once you have created a blank **System.Windows.Forms.MenuItem** using this constructor, you can use the properties and methods of the

**System.Windows.Forms.MenuItem** class to specify the appearance and behavior of your **System.Windows.Forms.MenuItem** .

*d) MenuItem*

*Example Syntax:*

*e) ToString*

[C#]                    public                    MenuItem(string                    text);

[C++]                    public:                    MenuItem(String\*                    text);

[VB]            Public            Sub            New(ByVal            text            As            String)

[JScript]            public            function            MenuItem(text            :            String);

*Description*

Initializes a new instance of the **System.Windows.Forms.MenuItem** class with a specified caption for the menu item.

When you specify a caption for your menu item with the *text* parameter, you can also specify an access key by placing an '&' character before the character to be used as the access key. For example, to specify the "F" in "File" as an access key, you would specify the caption for the menu item as "&File". You can use this feature to provide keyboard navigation for your menus. The caption for the menu item.

*f) MenuItem*

*Example Syntax:*

*g) ToString*

[C#]            public            MenuItem(string            text,            EventHandler            onClick);

[C++]            public:            MenuItem(String\*            text,            EventHandler\*            onClick);

[VB] Public Sub New(ByVal text As String, ByVal onClick As EventHandler)

[JScript] public function MenuItem(text : String, onClick : EventHandler);

## Description

Initializes a new instance of the class with a specified caption and event handler for the **System.Windows.Forms.MenuItem.Click** event of the menu item.

When you specify a caption for your menu item with the *text* parameter, you can also specify an access key by placing an '&' before the character to be used as the access key. For example, to specify the "F" in "File" as an access key, you would specify the caption for the menu item as "&File". You can use this feature to provide keyboard navigation for your menus. The caption for the menu item. The **System.EventHandler** that handles the **System.Windows.Forms.MenuItem.Click** event for this menu item.

### h) MenuItem

#### Example Syntax:

### i) ToString

```
[C#]      public      MenuItem(string      text,      MenuItem[]      items);
[C++]     public:     MenuItem(String*      text,      MenuItem*      items[]);
[VB]      Public Sub New(ByVal text As String, ByVal items() As MenuItem)
[JavaScript] public function MenuItem(text : String, items : MenuItem[]);
```

## Description

Initializes a new instance of the class with a specified caption and an array of submenu items defined for the menu item.

When you specify a caption for your menu item with the *text* parameter, you can also specify an access key by placing an '&' before the character to be used as the access key. For example, to specify the "F" in "File" as an access key, you would specify the caption for the menu item as "&File". You can use this feature to provide keyboard navigation for your menus. The caption for the menu item. An array of **System.Windows.Forms.MenuItem** objects that contains the submenu items for this menu item.

j) *MenuItem*

*Example Syntax:*

k) *ToString*

```
[C#] public MenuItem(string text, EventHandler onClick, Shortcut shortcut);  
[C++] public: MenuItem(String* text, EventHandler* onClick, Shortcut shortcut);  
[VB] Public Sub New(ByVal text As String, ByVal onClick As EventHandler,  
ByVal shortcut As Shortcut)  
[JScript] public function MenuItem(text : String, onClick : EventHandler, shortcut  
: Shortcut);
```

*Description*

Initializes a new instance of the class with a specified caption, event handler, and associated shortcut key for the menu item.

When you specify a caption for your menu item with the *text* parameter, you can also specify an access key by placing an '&' before the character to be used as the access key. For example, to specify the "F" in "File" as an access key, you would specify the caption for the menu item as "&File". You can use this feature to provide keyboard navigation for your menus. This constructor also enables you to specify a shortcut key in addition to an access key to provide keyboard navigation. Shortcut keys allow you to specify a combination of keys that can be used to activate the menu item. The caption for the menu item. The

**System.EventHandler** that handles the **System.Windows.Forms.MenuItem.Click** event for this menu item. One of the **System.Windows.Forms.Shortcut** values.

l) *MenuItem*

*Example Syntax:*

m) *ToString*

```
[C#] public MenuItem(MenuMerge mergeType, int mergeOrder, Shortcut
```

```

1 shortcut, string text, EventHandler onClick, EventHandler onPopup, EventHandler
2 onSelect, MenuItem[] items);
3 [C++] public: MenuItem(MenuMerge mergeType, int mergeOrder, Shortcut
4 shortcut, String* text, EventHandler* onClick, EventHandler* onPopup,
5 EventHandler* onSelect, MenuItem* items[]);
6 [VB] Public Sub New(ByVal mergeType As MenuMerge, ByVal mergeOrder As
7 Integer, ByVal shortcut As Shortcut, ByVal text As String, ByVal onClick As
8 EventHandler, ByVal onPopup As EventHandler, ByVal onSelect As
9 EventHandler, ByVal items() As MenuItem)
10 [JScript] public function MenuItem(mergeType : MenuMerge, mergeOrder : int,
11 shortcut : Shortcut, text : String, onClick : EventHandler, onPopup :
12 EventHandler, onSelect : EventHandler, items : MenuItem[]);

```

### *Description*

Initializes a new instance of the **System.Windows.Forms.MenuItem** class with a specified caption; defined event-handlers for the **System.Windows.Forms.MenuItem.Click**, **System.Windows.Forms.MenuItem.Select** and **System.Windows.Forms.MenuItem.Popup** events; a shortcut key; a merge type; and order specified for the menu item.

When you specify a caption for your menu item with the *text* parameter, you can also specify an access key by placing an '&' before the character to be used as the access key. For example, to specify the "F" in "File" as an access key, you would specify the caption for the menu item as "&File". You can use this feature to provide keyboard navigation for your menus. One of the **System.Windows.Forms.MenuMerge** values. The relative position that this menu item will assume in a merged menu. One of the **System.Windows.Forms.Shortcut** values. The caption for the menu item. The **System.EventHandler** that handles the **System.Windows.Forms.MenuItem.Click** event for this menu item. The **System.EventHandler** that handles the **System.Windows.Forms.MenuItem.Popup** event for this menu item. The **System.EventHandler** that handles the

**System.Windows.Forms.MenuItem.Select** event for this menu item. An array of **System.Windows.Forms.MenuItem** objects that contains the submenu items for this menu item.

*n) BarBreak*

*o) ToString*

[C#]            public            bool            BarBreak            {get;            set;}

[C++]   public:   \_\_property   bool   get\_BarBreak();public:   \_\_property   void  
set\_BarBreak(bool);

[VB]           Public           Property           BarBreak           As           Boolean

[JScript]   public   function   get   BarBreak() : Boolean;public   function   set  
BarBreak(Boolean);

### *Description*

Gets or sets a value indicating whether the **System.Windows.Forms.MenuItem** is placed on a new line (for a menu item added to a **System.Windows.Forms.MainMenu** object) or in a new column (for a submenu item or menu item displayed in a **System.Windows.Forms.ContextMenu** ).

You can use the **System.Windows.Forms.MenuItem.BarBreak** property to create a menu where each menu item is placed next to each other horizontally instead of in a vertical list. You can also use this property to create a menu bar that contains multiple rows of top-level menu items.

*p) Break*

*q) ToString*

[C#]            public            bool            Break            {get;            set;}

[C++]   public:   \_\_property   bool   get\_Break();public:   \_\_property   void  
set\_Break(bool);



```

1 [VB]          Public          Property          Break          As          Boolean
2 [JScript] public function get Break() : Boolean;public function set
3 Break(Boolean);
4

```

#### *Description*

Gets or sets a value indicating whether the item is placed on a new line (for a menu item added to a **System.Windows.Forms.MainMenu** object) or in a new column (for a menu item or submenu item displayed in a **System.Windows.Forms.ContextMenu** ).

You can use the **System.Windows.Forms.MenuItem.Break** property to create a menu where each menu is placed next to each other horizontally instead of in a vertical list. You can also use this property to create a menu bar that contains multiple rows of top-level menu items.

*r) Checked*

*s) ToString*

```

14 [C#]          public          bool          Checked          {get;          set;}
15 [C++] public: __property bool get_Checked();public: __property void
16 set_Checked(bool);
17

```

```

18 [VB]          Public          Property          Checked          As          Boolean
19 [JScript] public function get Checked() : Boolean;public function set
20 Checked(Boolean);
21

```

#### *Description*

Gets or sets a value indicating whether a check mark appears next to the text of the menu item.

You can use the **System.Windows.Forms.MenuItem.Checked** property in combination with other menu items in a menu to provide state for an application. For example, you can place a check mark on a menu item in a group of items to

identify the size of the font to be displayed for the text in an application. You can also use the **System.Windows.Forms.MenuItem.Checked** property to identify the selected menu item in a group of mutually exclusive menu items.

t) *Container*

u) *DefaultItem*

v) *ToString*

#### *Description*

Gets or sets a value indicating whether the menu item is the default menu item.

The default menu item for a menu is boldfaced. When the user double-clicks a submenu that contains a default item, the default item is selected, and the submenu is closed. You can use the

**System.Windows.Forms.MenuItem.DefaultItem** property to indicate, the default action that is expected in a menu or context menu.

w) *DesignMode*

x) *Enabled*

y) *ToString*

#### *Description*

Gets or sets a value indicating whether the menu item is enabled.

A **System.Windows.Forms.MenuItem** that is disabled is displayed in a gray color to indicate its state. When a parent menu item is disabled, all submenu items are not displayed.

- z) *Events*
- aa) *Handle*
- bb) *Index*
- cc) *ToString*

#### *Description*

Gets or sets a value indicating the position of the menu item in its parent menu.

This property provides the indexed position of a menu item in the menu item collection of its parent menu. You can use this property to reposition a menu item to a different location within its menu. You can also use this property when creating a **System.Windows.Forms.MenuItem** to specify its position in a menu structure at the time of creation.

dd) *IsParent*

ee) *ToString*

```
[C#]      public      override      bool      IsParent      {get;}
[C++]     public:     __property      virtual      bool      get_IsParent();
[VB]      Overrides   Public   ReadOnly   Property   IsParent   As   Boolean
[JScript] public      function      get      IsParent()      :      Boolean;
```

#### *Description*

Gets a value indicating whether the menu item contains child menu items.

You can use this property with the **System.Windows.Forms.MenuItem.Parent** property to navigate in code through an entire menu structure.

*ff) MdiList*

*gg) ToString*

[C#] public bool MdiList {get; set;}

[C++] public: \_\_property bool get\_MdiList();public: \_\_property void  
set\_MdiList(bool);

[VB] Public Property MdiList As Boolean

[JScript] public function get MdiList() : Boolean;public function set  
MdiList(Boolean);

### *Description*

Gets or sets a value indicating whether the menu item will be populated with a list of the Multiple Document Interface (MDI) child windows that are displayed within the associated form.

When a menu item is selected to display an MDI child window list, the list is displayed as a submenu of the menu item. Only forms that are defined as MDI child forms are displayed in the window list. Only nine child windows can be displayed at a time. If there are more than nine child windows displayed, a "More Windows..." menu item is displayed at the end of the window list. Clicking this menu item displays a dialog box with a complete list of the child windows that are currently active.

*hh) MdiListItem*

*ii) MenuID*

*jj) ToString*

### *Description*

Gets a value indicating the Windows identifier for this menu item.

*kk) MenuItem*

*ll) MergeOrder*

*mm) ToString*

*Description*

Gets or sets a value indicating the relative position of the menu item when it is merged with another.

The merge order of a menu item specifies the relative position that this menu item will assume if the menu structure that the **System.Windows.Forms.MenuItem** is contained in is merged with another.

*nn) MergeType*

*oo) ToString*

[C#]        public        MenuMerge        MergeType        {get;        set;}

[C++] public: \_\_property MenuMerge get\_MergeType();public: \_\_property void  
set\_MergeType(MenuMerge);

[VB]        Public        Property        MergeType        As        MenuMerge

[JScript] public function get MergeType() : MenuMerge;public function set  
MergeType(MenuMerge);

*Description*

Gets or sets a value indicating the behavior of this menu item when its menu is merged with another.

The merge type of a menu item indicates how the menu item behaves when it has the same merge order as another menu item being merged. You can use merged menus to create a consolidated menu based on two or more existing menus.

*pp) Mnemonic*

*qq) ToString*

```
[C#]          public          char          Mnemonic          {get;}
[C++]         public:         __property     __wchar_t         get_Mnemonic();
[VB]          Public          ReadOnly       Property          Mnemonic          As          Char
[JScript]     public          function       get               Mnemonic()       :          Char;
```

#### *Description*

Gets a value indicating the mnemonic character that is associated with this menu item.

The mnemonic character is the first character after an ampersand character in the text of the **System.Windows.Forms.MenuItem**. This property will not return a mnemonic if two ampersand characters are placed together as the ampersands are used to display an ampersand in the text of the **System.Windows.Forms.MenuItem** instead of defining a mnemonic character.

*rr) OwnerDraw*

*ss) ToString*

```
[C#]          public          bool          OwnerDraw          {get;          set;}
[C++]         public:         __property     bool               get_OwnerDraw();public:         __property     void
set_OwnerDraw(bool);
[VB]          Public          Property          OwnerDraw          As          Boolean
[JScript]     public          function       get               OwnerDraw()       : Boolean;public          function       set
OwnerDraw(Boolean);
```

#### *Description*

Gets or sets a value indicating whether the code that you provide draws the menu item or Windows draws the menu item.

When the **System.Windows.Forms.MenuItem.OwnerDraw** property is set to **true**, you need to handle all drawing of the menu item. You can use this capability to create your own special menu displays.

*tt) Parent*

*uu) ToString*

|           |         |            |              |                |
|-----------|---------|------------|--------------|----------------|
| [C#]      | public  | Menu       | Parent       | {get;}         |
| [C++]     | public: | __property | Menu*        | get_Parent();  |
| [VB]      | Public  | ReadOnly   | Property     | Parent As Menu |
| [JScript] | public  | function   | get Parent() | : Menu;        |

#### *Description*

Gets a value indicating the menu that contains this menu item.

You can use this property to obtain the **System.Windows.Forms.Menu** object for a submenu. You can cast the **System.Windows.Forms.Menu** object returned by this property to a **System.Windows.Forms.MenuItem** object to manipulate it.

*vv) RadioCheck*

*ww) ToString*

|           |         |            |                  |                                                                |
|-----------|---------|------------|------------------|----------------------------------------------------------------|
| [C#]      | public  | bool       | RadioCheck       | {get; set;}                                                    |
| [C++]     | public: | __property | bool             | get_RadioCheck();public: __property void set_RadioCheck(bool); |
| [VB]      | Public  | Property   | RadioCheck       | As Boolean                                                     |
| [JScript] | public  | function   | get RadioCheck() | : Boolean;public function set                                  |

1 RadioCheck(Boolean);

3 *Description*

4 Gets or sets a value indicating whether the  
5 **System.Windows.Forms.MenuItem** , if checked, displays a radio-button  
instead of a check mark.

6 Check marks do not necessarily imply a mutually exclusive state for a group of  
7 menu items. You can use this property to indicate to the user that the check  
mark of a menu item is mutually exclusive.

8       xx)    *Shortcut*

9       yy)    *ToString*

11 [C#]       public       Shortcut       Shortcut       {get;       set;}

12 [C++] public: \_\_property Shortcut get\_Shortcut();public: \_\_property void  
13 set\_Shortcut(Shortcut);

14 [VB]       Public       Property       Shortcut       As       Shortcut

15 [JScript] public function get Shortcut() : Shortcut;public function set  
16 Shortcut(Shortcut);

18 *Description*

19 Gets or sets a value indicating the shortcut key associated with the menu item.

20 Shortcut keys provide a method for users to activate frequently used menu items  
21 in your menu system and to provide keyboard access to your application for  
those users who do not have access to a mouse or other pointer device.



1           **zz) ShowShortcut**

2           **aaa) ToString**

3  
4 [C#]           public           bool           ShowShortcut           {get;           set;}

5 [C++] public: \_\_property bool get\_ShowShortcut();public: \_\_property void  
6 set\_ShowShortcut(bool);

7 [VB]           Public           Property           ShowShortcut           As           Boolean

8 [JScript] public function get ShowShortcut() : Boolean;public function set  
9 ShowShortcut(Boolean);

10  
11 *Description*

12 Gets or sets a value indicating whether the shortcut key that is associated with  
13 the menu item is displayed next to the menu item caption.

14 You can use this property to provide the option for users to hide shortcuts from  
15 menus to conserve menu space or to hide a shortcut key from being displayed.

16           **bbb) Site**

17           **ccc) Text**

18           **ddd) ToString**

19  
20 *Description*

21 Gets or sets a value indicating the caption of the menu item.

22 When you specify a caption for your menu item with the *text* parameter, you can  
23 also specify an access key by placing an '&' before the character to be used as  
24 the access key. For example, to specify the "F" in "File" as an access key, you  
25 would specify the caption for the menu item as "&File". You can use this feature  
to provide keyboard navigation for your menus.

eee) *Visible*

fff) *ToString*

[C#]            public            bool            Visible            {get;            set;}

[C++]   public:   \_\_property   bool   get\_Visible();public:   \_\_property   void  
set\_Visible(bool);

[VB]            Public            Property            Visible            As            Boolean

[JScript]   public   function   get   Visible()   :   Boolean;public   function   set  
Visible(Boolean);

#### *Description*

Gets or sets a value indicating whether the menu item is visible.

You can use this property to modify a menu structure without having to merge menus or disable menus. For example, if you want to hide a complete section of functionality from the menus for your application, you can hide them from the user by setting this property to **false** .

ggg) *ToString*

[C#]            public            event            EventHandler            Click;

[C++]            public:            \_\_event            EventHandler\*            Click;

[VB]            Public            Event            Click            As            EventHandler

#### *Description*

Occurs when the menu item is clicked or selected using a shortcut key or access key defined for the menu item.

The **System.Windows.Forms.MenuItem.Click** event occurs when this **System.Windows.Forms.MenuItem** is clicked by the user. This event also occurs if the user selects the menu item using the keyboard and presses the

Enter key. It can also occur if an access key or shortcut key is pressed that is associated with the **System.Windows.Forms.MenuItem** . For more information about handling events, see .

### *hhh) ToString*

#### *Description*

Occurs when the **System.Windows.Forms.MenuItem.OwnerDraw** property of a menu item is set to **true** and a request is made to draw the menu item.

The **System.Windows.Forms.DrawItemEventArgs** argument passed to a **System.Windows.Forms.MenuItem.DrawItem** event handler provides a **System.Drawing.Graphics** object that enables you to perform drawing and other graphical operations on the surface of the menu item. You can use this event handler to create custom menus that meet the needs of your application. For more information about handling events, see .

### *iii) ToString*

|       |         |         |                          |                         |
|-------|---------|---------|--------------------------|-------------------------|
| [C#]  | public  | event   | MeasureItemEventHandler  | MeasureItem;            |
| [C++] | public: | __event | MeasureItemEventHandler* | MeasureItem;            |
| [VB]  | Public  | Event   | MeasureItem As           | MeasureItemEventHandler |

#### *Description*

Occurs when the menu needs to know the size of a menu item before drawing it.

In order for this event to be raised, you must have the **System.Windows.Forms.MenuItem.OwnerDraw** property of the menu item set to **true** . This event is raised before owner drawn menus are drawn to allow for the size of the menu item to be drawn to be specified. For more information about handling events, see .

### *jjj) ToString*

|       |         |         |               |              |
|-------|---------|---------|---------------|--------------|
| [C#]  | public  | event   | EventHandler  | Popup;       |
| [C++] | public: | __event | EventHandler* | Popup;       |
| [VB]  | Public  | Event   | Popup As      | EventHandler |

#### *Description*

Occurs before a menu item's list of menu items is displayed.

This event only occurs when a menu item has submenu items to display. You can use this event handler to add, remove, enable, disable, check, or uncheck menu items based on the state of your application before they are displayed. For more information about handling events, see .

### *kkk) ToString*

|       |         |         |               |              |
|-------|---------|---------|---------------|--------------|
| [C#]  | public  | event   | EventHandler  | Select;      |
| [C++] | public: | __event | EventHandler* | Select;      |
| [VB]  | Public  | Event   | Select As     | EventHandler |

#### *Description*

Occurs when the user places the cursor over a menu item.

This event is typically raised when the user places the mouse cursor over the menu item. The event can also be raised when the user highlights a menu item using the keyboard by scrolling to the menu item with the arrow keys. You can use this event to display a detailed help string pertaining to this menu item in an application's status bar. For more information about handling events, see .

### *lll) CloneMenu*

|      |        |         |          |              |
|------|--------|---------|----------|--------------|
| [C#] | public | virtual | MenuItem | CloneMenu(); |
|------|--------|---------|----------|--------------|

```

1 [C++]      public:      virtual      MenuItem*      CloneMenu();
2 [VB]      Overridable  Public  Function  CloneMenu()  As  MenuItem
3 [JScript] public function CloneMenu() : MenuItem; Creates a copy of a
4 System.Windows.Forms.MenuItem

```

#### Description

Creates a copy of the current **System.Windows.Forms.MenuItem** .

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the duplicated menu item.

**System.Windows.Forms.MenuItem** objects cannot be used in more than one place unless you obtain a copy of the **System.Windows.Forms.MenuItem** . You can call this method to create a copy of this menu item for use in a **System.Windows.Forms.ContextMenu** , **System.Windows.Forms.MainMenu** , or other **System.Windows.Forms.MenuItem** within your application. When a menu item is cloned, any event handlers specified in the original menu item will continue to function in the cloned version of the menu item. For example, if you created a **System.Windows.Forms.MenuItem** and connected its **System.Windows.Forms.MenuItem.Click** event to an event handler. When the menu item is cloned, the cloned menu item will call the same event handler.

#### *mmm) CloneMenu*

```

17 [C#]      protected      void      CloneMenu(MenuItem      itemSrc);
18 [C++]      protected:      void      CloneMenu(MenuItem*      itemSrc);
19 [VB]      Protected  Sub  CloneMenu(ByVal  itemSrc  As  MenuItem)
20 [JScript]  protected  function  CloneMenu(itemSrc  :  MenuItem);

```

#### Description

Creates a copy of the specified **System.Windows.Forms.MenuItem** .

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the duplicated menu item.

Call this method to create copies of menu items that you have already created for use in a context menu or another menu structure within your application. This version of **System.Windows.Forms.MenuItem.CloneMenu** allows you to specify a specific **System.Windows.Forms.MenuItem** to copy instead of the menu item that is calling the method. You can use this method to initialize a new **System.Windows.Forms.MenuItem** object with a copy of another **System.Windows.Forms.MenuItem** . When a menu item is cloned, any event handlers specified in the original menu item will continue to function in the cloned version of the menu item. For example, if you created a **System.Windows.Forms.MenuItem** and connected its **System.Windows.Forms.MenuItem.Click** event to an event handler. When the menu item is cloned, the cloned menu item will call the same event handler. The **System.Windows.Forms.MenuItem** that represents the menu item to copy.

*nnn) Dispose*

```
[C#]      protected      override      void      Dispose(bool      disposing);
[C++]      protected:      void      Dispose(bool      disposing);
[VB]      Overrides      Protected      Sub      Dispose(ByVal      disposing      As      Boolean)
[JScript]      protected      override      function      Dispose(disposing      :      Boolean);
```

### *Description*

Disposes of the resources (other than memory) used by the **System.Windows.Forms.MenuItem** .

Call **System.Windows.Forms.MenuItem.Dispose(System.Boolean)** when you are finished using the **System.Windows.Forms.MenuItem** . The **System.Windows.Forms.MenuItem.Dispose(System.Boolean)** method leaves the **System.Windows.Forms.MenuItem** in an unusable state. After calling **System.Windows.Forms.MenuItem.Dispose(System.Boolean)** , you must release all references to the **System.Windows.Forms.MenuItem** so the memory it was occupying can be reclaimed by garbage collection.

*ooo) MergeMenu*

```
[C#]      public      virtual      MenuItem      MergeMenu();
```

```

1 [C++]      public:      virtual      MenuItem*      MergeMenu();
2 [VB]      Overridable  Public  Function  MergeMenu()  As  MenuItem
3 [JScript] public  function  MergeMenu()  :  MenuItem;  Merges  this
4 System.Windows.Forms.MenuItem          with          another
5 System.Windows.Forms.MenuItem          .

```

### Description

Merges this **System.Windows.Forms.MenuItem** with another **System.Windows.Forms.MenuItem** and returns the resulting merged **System.Windows.Forms.MenuItem** .

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the merged menu item.

When you call this version of **System.Windows.Forms.MenuItem.MergeMenu** , the **System.Windows.Forms.MenuItem** returned is a copy of the current menu item that can be merged with another menu item without affecting the functionality of the current item. This version of the **System.Windows.Forms.MenuItem.MergeMenu** method is similar to calling the **System.Windows.Forms.MenuItem.CloneMenu** method that contains no parameters.

*ppp) MergeMenu*

```

18 [C#]      public      new      void      MergeMenu(MenuItem      itemSrc);
19 [C++]      public:      void      MergeMenu(MenuItem*      itemSrc);
20 [VB]      Shadows  Public  Sub  MergeMenu(ByVal  itemSrc  As  MenuItem)
21 [JScript] public  hide  function  MergeMenu(itemSrc  :  MenuItem);

```

### Description

Merges another menu item with this menu item.

Menu items are merged according to the value of the menu item's **System.Windows.Forms.MenuItem.MergeType** and **System.Windows.Forms.MenuItem.MergeOrder** properties. This version of the **System.Windows.Forms.MenuItem.MergeMenu** method enables you to merge two **System.Windows.Forms.MenuItem** objects (and their submenus) into a single menu. Menu merging is handled automatically when a Multiple Document Interface(MDI) parent form and a child both have menus. You can use this version of the method to merge two **System.Windows.Forms.MenuItem** objects (and their submenu items) located in a **System.Windows.Forms.MainMenu** control into a single menu within a **System.Windows.Forms.ContextMenu** . For example, you can call this version of the **System.Windows.Forms.MenuItem.MergeMenu** method to merge the menu items of a File and Edit menu into a single **System.Windows.Forms.MenuItem** that can then be added to and displayed by a **System.Windows.Forms.ContextMenu** . A **System.Windows.Forms.MenuItem** that specifies the menu item to merge with this one.

*qqq) OnClick*

|           |             |           |                         |                               |
|-----------|-------------|-----------|-------------------------|-------------------------------|
| [C#]      | protected   | virtual   | void                    | OnClick(EventArgs e);         |
| [C++]     | protected:  | virtual   | void                    | OnClick(EventArgs* e);        |
| [VB]      | Overridable | Protected | Sub                     | OnClick(ByVal e As EventArgs) |
| [JScript] | protected   | function  | OnClick(e : EventArgs); |                               |

#### *Description*

Raises the **System.Windows.Forms.MenuItem.Click** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

*rrr) OnDrawItem*

|       |            |         |      |                                   |
|-------|------------|---------|------|-----------------------------------|
| [C#]  | protected  | virtual | void | OnDrawItem(DrawItemEventArgs e);  |
| [C++] | protected: | virtual | void | OnDrawItem(DrawItemEventArgs* e); |



1 [VB] Overridable Protected Sub OnDrawItem(ByVal e As DrawItemEventArgs)

2 [JScript] protected function OnDrawItem(e : DrawItemEventArgs);

3  
4 *Description*

5 Raises the **System.Windows.Forms.MenuItem.DrawItem** event.

6 Raising an event invokes the event handler through a delegate. For more  
7 information, see . A **System.Windows.Forms.DrawItemEventArgs** that  
contains the event data.

8 *sss) OnInitMenuPopup*

9  
10 [C#] protected virtual void OnInitMenuPopup(EventArgs e);

11 [C++] protected: virtual void OnInitMenuPopup(EventArgs\* e);

12 [VB] Overridable Protected Sub OnInitMenuPopup(ByVal e As EventArgs)

13 [JScript] protected function OnInitMenuPopup(e : EventArgs); An  
14 **System.EventArgs** that contains the event data.

15 *ttt) OnMeasureItem*

16  
17 [C#] protected virtual void OnMeasureItem(MeasureItemEventArgs e);

18 [C++] protected: virtual void OnMeasureItem(MeasureItemEventArgs\* e);

19 [VB] Overridable Protected Sub OnMeasureItem(ByVal e As  
20 MeasureItemEventArgs)

21 [JScript] protected function OnMeasureItem(e : MeasureItemEventArgs);

22  
23 *Description*

24 Raises the **System.Windows.Forms.MenuItem.MeasureItem** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.MeasureItemEventArgs** that contains the event data.

#### *uuu) OnPopup*

[C#]       protected       virtual       void       OnPopup(EventArgs e);

[C++]       protected:       virtual       void       OnPopup(EventArgs\* e);

[VB]   Overridable   Protected   Sub   OnPopup(ByVal e As EventArgs)

[JScript]       protected       function       OnPopup(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.MenuItem.Popup** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

#### *vvv) OnSelect*

[C#]       protected       virtual       void       OnSelect(EventArgs e);

[C++]       protected:       virtual       void       OnSelect(EventArgs\* e);

[VB]   Overridable   Protected   Sub   OnSelect(ByVal e As EventArgs)

[JScript]       protected       function       OnSelect(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.MenuItem.Select** event.

Raising an event invokes the event handler through a delegate. For more information, see . An **System.EventArgs** that contains the event data.

www) *PerformClick*

|           |         |          |                 |
|-----------|---------|----------|-----------------|
| [C#]      | public  | void     | PerformClick(); |
| [C++]     | public: | void     | PerformClick(); |
| [VB]      | Public  | Sub      | PerformClick()  |
| [JScript] | public  | function | PerformClick(); |

*Description*

Generates a **System.Windows.Forms.Control.Click** event for the **System.Windows.Forms.MenuItem** , simulating a click by a user.

You can use this menu to activate a menu item through code without passing any event information. For example, if you want to trigger a menu item based on an action that occurs in your application, you can call the **System.Windows.Forms.MenuItem.PerformClick** method for that **System.Windows.Forms.MenuItem** .

xxx) *PerformSelect*

|           |             |          |                  |                  |
|-----------|-------------|----------|------------------|------------------|
| [C#]      | public      | virtual  | void             | PerformSelect(); |
| [C++]     | public:     | virtual  | void             | PerformSelect(); |
| [VB]      | Overridable | Public   | Sub              | PerformSelect()  |
| [JScript] | public      | function | PerformSelect(); |                  |

*Description*

Raises the **System.Windows.Forms.MenuItem.Select** event for this menu item.

This method allows you to raise the **System.Windows.Forms.MenuItem.Select** event without passing any event information to the event handler.

yyy) *ToString*

|           |           |          |                     |             |
|-----------|-----------|----------|---------------------|-------------|
| [C#]      | public    | override | string              | ToString(); |
| [C++]     | public:   |          | String*             | ToString(); |
| [VB]      | Overrides | Public   | Function ToString() | As String   |
| [JScript] | public    | override | function ToString() | : String;   |

*Description*

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

Menu.MenuItemCollection class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a collection of **System.Windows.Forms.MenuItem** objects.

This class represents the collection of **System.Windows.Forms.MenuItem** objects stored in a **System.Windows.Forms.MainMenu** , **System.Windows.Forms.ContextMenu** , or **System.Windows.Forms.MenuItem** . For the **System.Windows.Forms.MainMenu** and **System.Windows.Forms.ContextMenu** classes, this collection represents the entire menu structure for the control. For the **System.Windows.Forms.MenuItem** class, this collection represents the list of submenu items associated with the **System.Windows.Forms.MenuItem** .

b) *Menu.MenuItemCollection*

*Example Syntax:*

### c) ToString

```

1
2 [C#]      public      Menu.MenuItemCollection(Menu      owner);
3
4 [C++]      public:      MenuItemCollection(Menu*      owner);
5
6 [VB]      Public      Sub      New(ByVal      owner      As      Menu)
7
8 [JScript] public      function      Menu.MenuItemCollection(owner      :      Menu);
9

```

### Description

Initializes a new instance of the **System.Windows.Forms.Menu.MenuItemCollection** class.

This class requires that you associate the collection with a class that derives from **System.Windows.Forms.Menu**, such as the **System.Windows.Forms.MainMenu**, **System.Windows.Forms.ContextMenu**, or **System.Windows.Forms.MenuItem** class. Since you must specify a menu that is associated with this collection, you cannot create multiple menu item collections and associate them with a menu as needed. In order to switch the menu items associated with a menu, you must clear the collection of items and add the menu items to display to the collection. The **System.Windows.Forms.Menu** that owns this collection.

### d) Count

### e) ToString

```

16
17
18
19 [C#]      public      int      Count      {get;}
20
21 [C++]      public:      __property      int      get_Count();
22
23 [VB]      Public      ReadOnly      Property      Count      As      Integer
24
25 [JScript] public      function      get      Count()      :      int;

```

### Description

Gets a value indicating the total number of **System.Windows.Forms.MenuItem** objects in the collection.

The **System.Windows.Forms.Menu.MenuItemCollection.Count** property holds the number of **System.Windows.Forms.MenuItem** objects assigned to the collection. You can use the **System.Windows.Forms.Menu.MenuItemCollection.Count** property value as the upper bounds of a loop to iterate through a collection. Keep in mind, the index value of a collection is a zero-based index, so you must subtract one from the looping variable. If you do not account for this, you will exceed the upper bounds of the collection and throw an exception.

f) *IsReadOnly*

g) *ToString*

|           |         |            |            |                         |
|-----------|---------|------------|------------|-------------------------|
| [C#]      | public  | bool       | IsReadOnly | {get;}                  |
| [C++]     | public: | __property | bool       | get_IsReadOnly();       |
| [VB]      | Public  | ReadOnly   | Property   | IsReadOnly As Boolean   |
| [JScript] | public  | function   | get        | IsReadOnly() : Boolean; |

#### *Description*

Gets a value indicating whether the collection is read-only.

h) *Item*

i) *ToString*

|           |             |            |                                       |                 |                                                   |
|-----------|-------------|------------|---------------------------------------|-----------------|---------------------------------------------------|
| [C#]      | public      | virtual    | MenuItem                              | this[int index] | {get;}                                            |
| [C++]     | public:     | __property | virtual                               | MenuItem*       | get_Item(int index);                              |
| [VB]      | Overridable | Public     | Default                               | ReadOnly        | Property Item(ByVal index As Integer) As MenuItem |
| [JScript] | returnValue | =          | MenuItemCollectionObject.Item(index); |                 |                                                   |

## Description

Retrieves the **System.Windows.Forms.MenuItem** at the specified indexed location in the collection.

To assign **System.Windows.Forms.MenuItem** objects to a specific location, or to retrieve them from the **System.Windows.Forms.Menu.MenuItemCollection**, you can reference the collection object with a specific index value. The index value of the **System.Windows.Forms.Menu.MenuItemCollection** is a zero-based index. The indexed location of the **System.Windows.Forms.MenuItem** in the collection.

### j) Add

```
[C#]      public      virtual      int      Add(MenuItem      item);
[C++]     public:     virtual      int      Add(MenuItem*      item);
[VB]      Overridable Public Function Add(ByVal item As MenuItem) As Integer
[JScript] public      function      Add(item      :      MenuItem)      :      int;
```

## Description

Adds a previously created **System.Windows.Forms.MenuItem** to the end of the current menu.

*Return Value:* The zero-based index where the item is stored in the collection.

A **System.Windows.Forms.MenuItem** can only be contained in one menu at a time, and may not be added more than once to the same menu. To reuse a **System.Windows.Forms.MenuItem** in more than one menu, use the **System.Windows.Forms.MenuItem.CloneMenu** method of the **System.Windows.Forms.MenuItem** class. To remove a **System.Windows.Forms.MenuItem** that you have previously added, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. The **System.Windows.Forms.MenuItem** to add.

k) *Add*

```
[C#]      public      virtual      MenuItem      Add(string      caption);
[C++]     public:     virtual      MenuItem*      Add(String*      caption);
[VB] Overridable Public Function Add(ByVal caption As String) As MenuItem
[JScript] public function Add(caption : String) : MenuItem; Adds a new
System.Windows.Forms.MenuItem to the collection.
```

*Description*

Adds a new **System.Windows.Forms.MenuItem** , to the end of the current menu, with a specified caption.

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the menu item being added to the collection.

A **System.Windows.Forms.MenuItem** can only be contained in one menu at a time, and may not be added more than once to the same menu. To reuse a **System.Windows.Forms.MenuItem** in more than one menu, use the **System.Windows.Forms.MenuItem.CloneMenu** method of the **System.Windows.Forms.MenuItem** class. To remove a **System.Windows.Forms.MenuItem** that you have previously added, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. The caption of the menu item.

l) *Add*

```
[C#]      public      virtual      int      Add(int      index,      MenuItem      item);
[C++]     public:     virtual      int      Add(int      index,      MenuItem*      item);
[VB] Overridable Public Function Add(ByVal index As Integer, ByVal item As
MenuItem) As Integer
[JScript] public function Add(index : int, item : MenuItem) : int;
```



## Description

Adds a previously created **System.Windows.Forms.MenuItem** at the specified index within the menu item collection.

*Return Value:* The zero-based index where the item is stored in the collection.

A **System.Windows.Forms.MenuItem** can only be contained in one menu at a time, and may not be added more than once to the same menu. To reuse a **System.Windows.Forms.MenuItem** in more than one menu, use the **System.Windows.Forms.MenuItem.CloneMenu** method of the **System.Windows.Forms.MenuItem** class. To remove a **System.Windows.Forms.MenuItem** that you have previously added, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. The position to add the new item. The **System.Windows.Forms.MenuItem** to add.

### m) Add

```
[C#] public virtual MenuItem Add(string caption, EventHandler onClick);
```

```
[C++] public: virtual MenuItem* Add(String* caption, EventHandler* onClick);
```

```
[VB] Overridable Public Function Add(ByVal caption As String, ByVal onClick  
As EventHandler) As MenuItem
```

```
[JScript] public function Add(caption : String, onClick : EventHandler) :  
MenuItem;
```

## Description

Adds a new **System.Windows.Forms.MenuItem** to the end of the current menu with a specified caption and a specified event handler for the **System.Windows.Forms.MenuItem.Click** event.

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the menu item being added to the collection.

A **System.Windows.Forms.MenuItem** can only be contained in one menu at a time, and may not be added more than once to the same menu. To reuse a **System.Windows.Forms.MenuItem** in more than one menu, use the **System.Windows.Forms.MenuItem.CloneMenu** method of the

**System.Windows.Forms.MenuItem** class. To remove a **System.Windows.Forms.MenuItem** that you have previously added, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. The caption of the menu item. An **System.EventHandler** that represents the event handler that is called when the item is clicked by the user or when a user, presses an accelerator or shortcut key for the menu item.

*n) Add*

[C#] public virtual MenuItem Add(string caption, MenuItem[] items);

[C++] public: virtual MenuItem\* Add(String\* caption, MenuItem\* items[]);

[VB] Overridable Public Function Add(ByVal caption As String, ByVal items()

As MenuItem) As MenuItem

[JScript] public function Add(caption : String, items : MenuItem[]) : MenuItem;

*Description*

Adds a new **System.Windows.Forms.MenuItem** to the end of this menu with the specified caption, **System.Windows.Forms.MenuItem.Click** event handler, and items.

*Return Value:* A **System.Windows.Forms.MenuItem** that represents the menu item being added to the collection.

A **System.Windows.Forms.MenuItem** can only be contained in one menu at a time, and may not be added more than once to the same menu. To reuse a **System.Windows.Forms.MenuItem** in more than one menu, use the **System.Windows.Forms.MenuItem.CloneMenu** method of the **System.Windows.Forms.MenuItem** class. To remove a **System.Windows.Forms.MenuItem** that you have previously added, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. The caption of the menu item. An array of **System.Windows.Forms.MenuItem** objects that this **System.Windows.Forms.MenuItem** will contain.

o) *AddRange*

```
[C#]      public      virtual      void      AddRange(MenuItem[]      items);
[C++]     public:     virtual      void      AddRange(MenuItem*      items[]);
[VB]      Overridable Public Sub      AddRange(ByVal items() As MenuItem)
[JScript] public      function      AddRange(items      :      MenuItem[]);
```

*Description*

Adds an array of previously created **System.Windows.Forms.MenuItem** objects to the collection.

You can use method to quickly add a group of previously created **System.Windows.Forms.MenuItem** objects to the collection instead of manually adding each **System.Windows.Forms.MenuItem** to the collection using the **System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)** method. If the collection already contains **System.Windows.Forms.MenuItem** objects, calling this method will add the new **System.Windows.Forms.MenuItem** objects to the end of the collection. An array of **System.Windows.Forms.MenuItem** objects representing the menu items to add to the collection.

p) *Clear*

```
[C#]          public          virtual          void          Clear();
[C++]         public:         virtual          void          Clear();
[VB]          Overridable          Public          Sub          Clear()
[JScript]          public          function          Clear();
```

*Description*

Removes all **System.Windows.Forms.MenuItem** objects from the menu item collection.

You can use this method to clear the entire collection of menu items from a menu. To remove an individual menu item from the collection, use the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method. To add new **System.Windows.Forms.MenuItem** objects to the collection, use the **System.Windows.Forms.Menu.MenuItemCollection.Add(System.String)** method.

#### q) *Contains*

```
[C#]      public      bool      Contains(MenuItem      value);
[C++]     public:      bool      Contains(MenuItem*      value);
[VB]      Public Function Contains(ByVal value As MenuItem) As Boolean
[JScript] public function Contains(value : MenuItem) : Boolean;
```

#### *Description*

Determines if the specified **System.Windows.Forms.MenuItem** is a member of the collection.

*Return Value:* **true** if the **System.Windows.Forms.MenuItem** is a member of the collection; otherwise, **false**.

This method enables you to determine whether a **System.Windows.Forms.MenuItem** is member of the collection before attempting to perform operations on the **System.Windows.Forms.MenuItem**. You can use this method to confirm that a **System.Windows.Forms.MenuItem** has been added to or is still a member of the collection. The **System.Windows.Forms.MenuItem** to locate in the collection.

#### r) *CopyTo*

```
[C#]      public      void      CopyTo(Array      dest,      int      index);
[C++]     public:      __sealed void CopyTo(Array*      dest,      int      index);
[VB]      NotOverridable Public Sub CopyTo(ByVal dest As Array, ByVal index As
```

Integer)

[JScript] public function CopyTo(dest : Array, index : int);

#### Description

Copies the entire collection into an existing array at a specified location within the array.

You can use this method to combine **System.Windows.Forms.MenuItem** objects from multiple collections into a single array. This feature enables you to easily combine two or more sets of menu items for use in a **System.Windows.Forms.ContextMenu** or **System.Windows.Forms.MainMenu**. The destination array. The index in the destination array at which storing begins.

#### s) GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: \_\_sealed IEnumerator\* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

#### Description

Returns an enumerator that can be used to iterate through the menu item collection.

*Return Value:* An **System.Collections.IEnumerator** object that represents the menu item collection.

#### t) IndexOf

[C#] public int IndexOf(MenuItem value);

[C++] public: int IndexOf(MenuItem\* value);

[VB] Public Function IndexOf(ByVal value As MenuItem) As Integer

[JScript] public function IndexOf(value : MenuItem) : int;

### Description

Retrieves the index of a specific item in the collection.

**Return Value:** The zero-based index of the item found in the collection; otherwise, -1. The **System.Windows.Forms.MenuItem** to locate in the collection.

### u) Remove

[C#] public virtual void Remove(MenuItem item);

[C++] public: virtual void Remove(MenuItem\* item);

[VB] Overridable Public Sub Remove(ByVal item As MenuItem)

[JScript] public function Remove(item : MenuItem);

### Description

Removes the specified **System.Windows.Forms.MenuItem** from the menu item collection.

When a **System.Windows.Forms.MenuItem** is removed from the menu item collection, all subsequent menu items are moved up one position in the collection. You can use this version of the **System.Windows.Forms.Menu.MenuItemCollection.Remove(System.Windows.Forms.MenuItem)** method to remove a specific **System.Windows.Forms.MenuItem** from the collection using a reference to the **System.Windows.Forms.MenuItem** to be removed. If you do not have a reference to the **System.Windows.Forms.MenuItem** that you want to remove, you can use the other version of this method that accepts, as a parameter, an index corresponding to the **System.Windows.Forms.MenuItem** to be removed. The **System.Windows.Forms.MenuItem** to remove.

v) *RemoveAt*

```
[C#]      public      virtual      void      RemoveAt(int      index);
[C++]     public:     virtual      void      RemoveAt(int      index);
[VB]      Overridable Public Sub RemoveAt(ByVal index As Integer)
[JavaScript] public function RemoveAt(index : int); Removes a
System.Windows.Forms.MenuItem from the menu item collection.
```

*Description*

Removes a **System.Windows.Forms.MenuItem** from the menu item collection at a specified index.

When a **System.Windows.Forms.MenuItem** is removed from the menu item collection, all subsequent menu items are moved up one position in the collection. The index of the **System.Windows.Forms.MenuItem** to remove.

w) *IList.Add*

```
[C#]      int      IList.Add(object      value);
[C++]     int      IList::Add(Object*      value);
[VB]      Function Add(ByVal value As Object) As Integer Implements IList.Add
[JavaScript] function IList.Add(value : Object) : int;
```

x) *IList.Contains*

```
[C#]      bool      IList.Contains(object      value);
[C++]     bool      IList::Contains(Object*      value);
[VB]      Function Contains(ByVal value As Object) As Boolean Implements
```

1 IList.Contains

2 [JScript] function IList.Contains(value : Object) : Boolean;

3 y) *IList.IndexOf*

4  
5 [C#] int IList.IndexOf(object value);

6 [C++] int IList::IndexOf(Object\* value);

7 [VB] Function IndexOf(ByVal value As Object) As Integer Implements

8 IList.IndexOf

9 [JScript] function IList.IndexOf(value : Object) : int;

10 z) *IList.Insert*

11  
12 [C#] void IList.Insert(int index, object value);

13 [C++] void IList::Insert(int index, Object\* value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17 aa) *IList.Remove*

18  
19 [C#] void IList.Remove(object value);

20 [C++] void IList::Remove(Object\* value);

21 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

22 [JScript] function IList.Remove(value : Object);



MenuMerge enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies the behavior of a **System.Windows.Forms.MenuItem** when it is merged with items in another menu.

Use the members of this enumeration to set the value of the **System.Windows.Forms.MenuItem.MergeType** property of the **System.Windows.Forms.MenuItem** .

*b) ToString*

|           |         |       |           |            |
|-----------|---------|-------|-----------|------------|
| [C#]      | public  | const | MenuMerge | Add;       |
| [C++]     | public: | const | MenuMerge | Add;       |
| [VB]      | Public  | Const | Add As    | MenuMerge  |
| [JScript] | public  | var   | Add :     | MenuMerge; |

*Description*

The **System.Windows.Forms.MenuItem** is added to the collection of existing **System.Windows.Forms.MenuItem** objects in a merged menu.

*c) ToString*

|           |         |       |               |             |
|-----------|---------|-------|---------------|-------------|
| [C#]      | public  | const | MenuMerge     | MergeItems; |
| [C++]     | public: | const | MenuMerge     | MergeItems; |
| [VB]      | Public  | Const | MergeItems As | MenuMerge   |
| [JScript] | public  | var   | MergeItems :  | MenuMerge;  |

*Description*

All submenu items of this **System.Windows.Forms.MenuItem** are merged with those of existing **System.Windows.Forms.MenuItem** objects at the same position in a merged menu.

*d) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuMerge | Remove;      |
| [C++]     | public: | const | MenuMerge | Remove;      |
| [VB]      | Public  | Const | Remove    | As MenuMerge |
| [JScript] | public  | var   | Remove    | : MenuMerge; |

*Description*

The **System.Windows.Forms.MenuItem** is not included in a merged menu.

*e) ToString*

|           |         |       |           |              |
|-----------|---------|-------|-----------|--------------|
| [C#]      | public  | const | MenuMerge | Replace;     |
| [C++]     | public: | const | MenuMerge | Replace;     |
| [VB]      | Public  | Const | Replace   | As MenuMerge |
| [JScript] | public  | var   | Replace   | : MenuMerge; |

*Description*

The **System.Windows.Forms.MenuItem** replaces an existing **System.Windows.Forms.MenuItem** at the same position in a merged menu.

1 Message structure (System.Windows.Forms)

2 a) *ToString*

3  
4  
5 *Description*

6 Implements a Windows message.

7 The **System.Windows.Forms.Message** class wraps messages that Windows  
8 sends. You can use this class to wrap a message and assign it to the window  
9 procedure to be dispatched. You can also use this class to get information about  
10 a message the system sends to your application or controls.

11 b) *HWND*

12 c) *ToString*

13 [C#] public IntPtr HWND {get; set;}

14 [C++] public: \_\_property IntPtr get\_HWnd();public: \_\_property void  
15 set\_HWnd(IntPtr);

16 [VB] Public Property HWND As IntPtr

17 [JScript] public function get HWND() : IntPtr;public function set HWND(IntPtr);

18  
19 *Description*

20 Gets or sets the window handle of the message.

21 d) *LParam*

22 e) *ToString*

23  
24 [C#] public IntPtr LParam {get; set;}

25 [C++] public: \_\_property IntPtr get\_LParam();public: \_\_property void

1 set\_LParam(IntPtr);

2 [VB] Public Property LParam As IntPtr

3 [JScript] public function get LParam() : IntPtr;public function set LParam(IntPtr);

4  
5 *Description*

6 Specifies the **System.Windows.Forms.Message.LParam** field of the message.

7 The value of this field depends on the message. Use the  
8 **System.Windows.Forms.Message.LParam** field to get information that is important for handling the message.

9  
10 *f) Msg*

11 *g) ToString*

12  
13 [C#] public int Msg {get; set;}

14 [C++] public: \_\_property int get\_Msg();public: \_\_property void set\_Msg(int);

15 [VB] Public Property Msg As Integer

16 [JScript] public function get Msg() : int;public function set Msg(int);

17  
18 *Description*

19 Gets or sets the ID number for the message.

20 *h) Result*

21 *i) ToString*

22  
23 [C#] public IntPtr Result {get; set;}

24 [C++] public: \_\_property IntPtr get\_Result();public: \_\_property void

25 set\_Result(IntPtr);

1 [VB] Public Property Result As IntPtr

2 [JScript] public function get Result() : IntPtr;public function set Result(IntPtr);

4 *Description*

5 Specifies the value that is returned to Windows in response to handling the  
6 message.

7 j) *WParam*

8 k) *ToString*

9 [C#] public IntPtr WParam {get; set;}

10 [C++] public: \_\_property IntPtr get\_WParam();public: \_\_property void  
11 set\_WParam(IntPtr);

12 [VB] Public Property WParam As IntPtr

13 [JScript] public function get WParam() : IntPtr;public function set  
14 WParam(IntPtr);

15  
16 *Description*

17 Gets or sets the **System.Windows.Forms.Message.WParam** field of the  
18 message.

19 The value of this field depends on the message. Use the  
20 **System.Windows.Forms.Message.WParam** field to get information that is  
21 important to handling the message. This field is typically used to store small  
22 pieces of information, such as flags.

23 l) *Create*

24 [C#] public static Message Create(IntPtr hWnd, int msg, IntPtr wparam, IntPtr  
25 lparam);

```

1 [C++] public: static Message Create(IntPtr hWnd, int msg, IntPtr wParam, IntPtr
2 lParam);
3 [VB] Public Shared Function Create(ByVal hWnd As IntPtr, ByVal msg As
4 Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Message
5 [JScript] public static function Create(hWnd : IntPtr, msg : int, wParam : IntPtr,
6 lParam          :          IntPtr)          :          Message;
7

```

### Description

Creates a new **System.Windows.Forms.Message** object.

*Return Value:* A **System.Windows.Forms.Message** object that represents the message that was created.

Use the

**System.Windows.Forms.Message.Create(System.IntPtr, System.Int32, System.IntPtr, System.IntPtr)** method to create a **System.Windows.Forms.Message** object to wrap a message sent by Windows. The window handle for the message. The message ID. The message wParam field. The message lParam field.

### m) Equals

```

17 [C#]      public      override      bool      Equals(object      o);
18 [C++]      public:      bool      Equals(Object*      o);
19 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean
20 [JScript] public override function Equals(o : Object) : Boolean;
21

```

### Description

n) *GetHashCode*

```
[C#]          public          override          int          GetHashCode();
[C++]          public:          int          GetHashCode();
[VB]  Overrides  Public  Function  GetHashCode()  As  Integer
[JScript]  public  override  function  GetHashCode()  :  int;
```

*Description*

o) *GetLParam*

```
[C#]          public          object          GetLParam(Type          cls);
[C++]          public:          Object*          GetLParam(Type*          cls);
[VB]  Public  Function  GetLParam(ByVal  cls  As  Type)  As  Object
[JScript]  public  function  GetLParam(cls  :  Type)  :  Object;
```

*Description*

Gets the **System.Windows.Forms.Message.LParam** value, and converts the value to an object.

*Return Value:* An **System.Object** that represents an instance of the class specified by the *cls* parameter, with the data from the **System.Windows.Forms.Message.LParam** field of the message.

Use the **System.Windows.Forms.Message.GetLParam(System.Type)** method to retrieve information from the **System.Windows.Forms.Message.LParam** field of a message and convert it to an object. You can use this method to access objects passed in a message. The class to use to create an instance. You must declare this class using the **@dll.struct** directive.

*p) ToString*

```
[C#]      public      override      string      ToString();
[C++]      public:      String*      ToString();
[VB]      Overrides      Public      Function      ToString()      As      String
[JScript]      public      override      function      ToString()      :      String;
```

*Description*

MessageBox class (System.Windows.Forms)

*a) ToString*

*Description*

Displays a message box that can contain text, buttons, and symbols that inform and instruct the user.

You cannot create a new instance of the

**System.Windows.Forms.MessageBox** class. To display a message box, call the **static** method

**System.Windows.Forms.MessageBox.Show(System.String, System.String, System.Windows.Forms.MessageBoxButtons, System.Windows.Forms.MessageBoxIcon, System.Windows.Forms.MessageBoxDefaultButton, System.Windows.Forms.MessageBoxOptions)** . The title, message, buttons, and icons displayed in the message box are determined by parameters that you pass to this method.

*b) Show*

```
[C#]      public      static      DialogResult      Show(string      text);
[C++]      public:      static      DialogResult      Show(String*      text);
[VB]      Public      Shared      Function      Show(ByVal      text      As      String)      As      DialogResult
```



1 [JScript] public static function Show(text : String) : DialogResult;

3 *Description*

4 Displays a message box with specified text.

5 *Return Value:* One of the **System.Windows.Forms.DialogResult** values.

6 By default, the message box displays an **OK** button. The message box does not contain a caption in the title. The text to display in the message box.

7 c) *Show*

9 [C#] public static DialogResult Show(IWin32Window owner, string text);

10 [C++] public: static DialogResult Show(IWin32Window\* owner, String\* text);

11 [VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text

12 As String) As DialogResult

13 [JScript] public static function Show(owner : IWin32Window, text : String) :

14 DialogResult;

16 *Description*

17 Displays a message box in front of the specified object and with the specified text.

18 *Return Value:* One of the **System.Windows.Forms.DialogResult** values.

19 You can use the *owner* parameter to specify a particular object, which implements the **System.Windows.Forms.IWin32Window** interface, to place the message box in front of. A message box is a modal dialog, which means no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (typically in response to some user action) before input to another form can occur. The **System.Windows.Forms.IWin32Window** the message box will display in front of. The text to display in the message box.

d) *Show*

```
[C#] public static DialogResult Show(string text, string caption);  
[C++] public: static DialogResult Show(String* text, String* caption);  
[VB] Public Shared Function Show(ByVal text As String, ByVal caption As  
String) As DialogResult  
[JScript] public static function Show(text : String, caption : String) : DialogResult;
```

*Description*

Displays a message box with specified text and caption.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

By default, the message box displays an **OK** button. The text to display in the message box. The text to display in the title bar of the message box.

e) *Show*

```
[C#] public static DialogResult Show(IWin32Window owner, string text, string  
caption);  
[C++] public: static DialogResult Show(IWin32Window* owner, String* text,  
String* caption);  
[VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text  
As String, ByVal caption As String) As DialogResult  
[JScript] public static function Show(owner : IWin32Window, text : String,  
caption : String) : DialogResult;
```

*Description*

Displays a message box in front of the specified object and with the specified text and caption.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can use the *owner* parameter to specify a particular object, which implements the **System.Windows.Forms.IWin32Window** interface, to place the message box in front of. A message box is a modal dialog, which means no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (typically in response to some user action) before input to another form can occur. The **System.Windows.Forms.IWin32Window** the message box will display in front of. The text to display in the message box. The text to display in the title bar of the message box.

#### f) *Show*

```
[C#] public static DialogResult Show(string text, string caption,
MessageBoxButtons buttons);
```

```
[C++] public: static DialogResult Show(String* text, String* caption,
MessageBoxButtons buttons);
```

```
[VB] Public Shared Function Show(ByVal text As String, ByVal caption As
String, ByVal buttons As MessageBoxButtons) As DialogResult
```

```
[JScript] public static function Show(text : String, caption : String, buttons :
MessageBoxButtons) : DialogResult;
```

#### *Description*

Displays a message box with specified text, caption, and buttons.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can have a maximum of three buttons on the message box. The text to display in the message box. The text to display in the title bar of the message box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies which buttons to display in the message box.

### g) *Show*

```
[C#] public static DialogResult Show(IWin32Window owner, string text, string
caption,                      MessageBoxButtons                      buttons);
[C++] public: static DialogResult Show(IWin32Window* owner, String* text,
String*                      caption,                      MessageBoxButtons                      buttons);
[VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text
As String, ByVal caption As String, ByVal buttons As MessageBoxButtons) As
DialogResult
[JavaScript] public static function Show(owner : IWin32Window, text : String,
caption : String, buttons : MessageBoxButtons) : DialogResult;
```

### *Description*

Displays a message box in front of the specified object and with the specified text, caption, and buttons.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can use the *owner* parameter to specify a particular object, which implements the **System.Windows.Forms.IWin32Window** interface, to place the message box in front of. A message box is a modal dialog, which means no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (typically in response to some user action) before input to another form can occur. The **System.Windows.Forms.IWin32Window** the message box will display in front of. The text to display in the message box. The text to display in the title bar of the message box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies which buttons to display in the message box.

### h) *Show*

```
[C#] public static DialogResult Show(string text, string caption,
```

```

1 MessageBoxButtons buttons, MessageBoxIcon icon);
2 [C++] public: static DialogResult Show(String* text, String* caption,
3 MessageBoxButtons buttons, MessageBoxIcon icon);
4 [VB] Public Shared Function Show(ByVal text As String, ByVal caption As
5 String, ByVal buttons As MessageBoxButtons, ByVal icon As MessageBoxIcon)
6 As DialogResult
7 [JScript] public static function Show(text : String, caption : String, buttons :
8 MessageBoxButtons, icon : MessageBoxIcon) : DialogResult;
9

```

### *Description*

Displays a message box with specified text, caption, buttons, and icon.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can have a maximum of three buttons on the message box. The text to display in the message box. The text to display in the title bar of the message box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies which buttons to display in the message box. One of the **System.Windows.Forms.MessageBoxIcon**; values that specifies which icon to display in the message box.

#### *i) Show*

```

18 [C#] public static DialogResult Show(IWin32Window owner, string text, string
19 caption, MessageBoxIcon buttons, MessageBoxIcon icon);
20 [C++] public: static DialogResult Show(IWin32Window* owner, String* text,
21 String* caption, MessageBoxIcon buttons, MessageBoxIcon icon);
22 [VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text
23 As String, ByVal caption As String, ByVal buttons As MessageBoxButtons,
24 ByVal icon As MessageBoxIcon) As DialogResult
25

```

1 [JScript] public static function Show(owner : IWin32Window, text : String,  
2 caption : String, buttons : MessageBoxButtons, icon : MessageBoxIcon) :  
3 DialogResult;

#### 5 *Description*

6 Displays a message box in front of the specified object and with the specified  
7 text, caption, buttons, and icon.

7 *Return Value:* One of the **System.Windows.Forms.DialogResult** values.

8 You can use the *owner* parameter to specify a particular object, Which  
9 implements the **System.Windows.Forms.IWin32Window** interface, to place  
10 the message box in front of. A message box is a modal dialog, which means no  
11 input (keyboard or mouse click) can occur except to objects on the modal form.  
12 The program must hide or close a modal form (typically in response to some  
13 user action) before input to another form can occur. The

11 **System.Windows.Forms.IWin32Window** the message box will display in  
12 front of. The text to display in the message box. The text to display in the title  
13 bar of the message box. One of the  
14 **System.Windows.Forms.MessageBoxButtons** that specifies which buttons  
15 to display in the message box. One of the  
16 **System.Windows.Forms.MessageBoxIcon** values that specifies which icon  
17 to display in the message box.

#### 16 *j) Show*

18 [C#] public static DialogResult Show(string text, string caption,  
19 MessageBoxButtons buttons, MessageBoxIcon icon, MessageBoxDefaultButton  
20 defaultButton);

21 [C++] public: static DialogResult Show(String\* text, String\* caption,  
22 MessageBoxButtons buttons, MessageBoxIcon icon, MessageBoxDefaultButton  
23 defaultButton);

24 [VB] Public Shared Function Show(ByVal text As String, ByVal caption As  
25 String, ByVal buttons As MessageBoxButtons, ByVal icon As MessageBoxIcon,

```

1 ByVal defaultButton As MessageBoxDefaultButton) As DialogResult
2 [JScript] public static function Show(text : String, caption : String, buttons :
3 MessageBoxButtons, icon : MessageBoxIcon, defaultButton :
4 MessageBoxDefaultButton) : DialogResult;

```

### 6 *Description*

7 Displays a message box with the specified text, caption, buttons, icon, and  
 8 default button.

8 *Return Value:* One of the **System.Windows.Forms.DialogResult** values.

9 You can have a maximum of three buttons on the message box. The text to  
 10 display in the message box. The text to display in the title bar of the message  
 11 box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies  
 12 which buttons to display in the message box. One of the  
 13 **System.Windows.Forms.MessageBoxIcon** values that specifies which icon  
 14 to display in the message box. One fo the  
 15 **System.Windows.Forms.MessageBoxDefaultButton** values which specifies  
 16 which is the default button for the message box.

### 14 *k) Show*

```

16 [C#] public static DialogResult Show(IWin32Window owner, string text, string
17 caption, MessageBoxButtons buttons, MessageBoxIcon icon,
18 MessageBoxDefaultButton defaultButton);

```

```

19 [C++] public: static DialogResult Show(IWin32Window* owner, String* text,
20 String* caption, MessageBoxButtons buttons, MessageBoxIcon icon,
21 MessageBoxDefaultButton defaultButton);

```

```

22 [VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text
23 As String, ByVal caption As String, ByVal buttons As MessageBoxButtons,
24 ByVal icon As MessageBoxIcon, ByVal defaultButton As
25 MessageBoxDefaultButton) As DialogResult

```

```

1 [JScript] public static function Show(owner : IWin32Window, text : String,
2 caption : String, buttons : MessageBoxButtons, icon : MessageBoxIcon,
3 defaultButton : MessageBoxButtons : DialogResult;
4

```

### 5 *Description*

6 Displays a message box in front of the specified object and with the specified text, caption, buttons, icon, and default button.

7 *Return Value:* One of the **System.Windows.Forms.DialogResult** values.

8 You can use the *owner* parameter to specify a particular object, which implements the **System.Windows.Forms.IWin32Window** interface, to place the message box in front of. A message box is a modal dialog, which means no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (typically in response to some user action) before input to another form can occur. The **System.Windows.Forms.IWin32Window** the message box will display in front of. The text to display in the message box. The text to display in the title bar of the message box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies which buttons to display in the message box. One of the **System.Windows.Forms.MessageBoxIcon** values that specifies which icon to display in the message box. One fo the **System.Windows.Forms.MessageBoxDefaultButton** values which specifies which is the default button for the message box.

### 17 *1) Show*

```

18
19 [C#] public static DialogResult Show(string text, string caption,
20 MessageBoxButtons buttons, MessageBoxIcon icon, MessageBoxButtons
21 defaultButton, MessageBoxButtons options);
22 [C++] public: static DialogResult Show(String* text, String* caption,
23 MessageBoxButtons buttons, MessageBoxIcon icon, MessageBoxButtons
24 defaultButton, MessageBoxButtons options);
25 [VB] Public Shared Function Show(ByVal text As String, ByVal caption As

```



```

1 String, ByVal buttons As MessageBoxButtons, ByVal icon As MessageBoxIcon,
2 ByVal defaultButton As MessageBoxDefaultButton, ByVal options As
3 MessageBoxOptions) As DialogResult
4 [JScript] public static function Show(text : String, caption : String, buttons :
5 MessageBoxButtons, icon : MessageBoxIcon, defaultButton :
6 MessageBoxDefaultButton, options : MessageBoxOptions) : DialogResult;
7 Displays a message box.

```

#### 9 *Description*

10 Displays a message box with the specified text, caption, buttons, icon, default  
 11 button, and options.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

12 You can have a maximum of three buttons on the message box. The text to  
 13 display in the message box. The text to display in the title bar of the message  
 14 box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies  
 15 which buttons to display in the message box. One of the  
 16 **System.Windows.Forms.MessageBoxIcon** values that specifies which icon  
 17 to display in the message box. One fo the  
 18 **System.Windows.Forms.MessageBoxDefaultButton** values which specifies  
 19 which is the default button for the message box. One of the  
 20 **System.Windows.Forms.MessageBoxOptions** values which specifies which  
 21 display and association options will be used for the message box.

#### 18 *m) Show*

```

20 [C#] public static DialogResult Show(IWin32Window owner, string text, string
21 caption, MessageBoxButtons buttons, MessageBoxIcon icon,
22 MessageBoxDefaultButton defaultButton, MessageBoxOptions options);
23 [C++] public: static DialogResult Show(IWin32Window* owner, String* text,
24 String* caption, MessageBoxButtons buttons, MessageBoxIcon icon,
25 MessageBoxDefaultButton defaultButton, MessageBoxOptions options);

```

```

1 [VB] Public Shared Function Show(ByVal owner As IWin32Window, ByVal text
2 As String, ByVal caption As String, ByVal buttons As MessageBoxButtons,
3 ByVal icon As MessageBoxIcon, ByVal defaultButton As
4 MessageBoxDefaultButton, ByVal options As MessageBoxOptions) As
5 DialogResult
6 [JScript] public static function Show(owner : IWin32Window, text : String,
7 caption : String, buttons : MessageBoxButtons, icon : MessageBoxIcon,
8 defaultButton : MessageBoxDefaultButton, options : MessageBoxOptions) :
9 DialogResult;

```

### *Description*

Displays a message box in front of the specified object and with the specified text, caption, buttons, icon, default button, and options.

*Return Value:* One of the **System.Windows.Forms.DialogResult** values.

You can use the *owner* parameter to specify a particular object, which implements the **System.Windows.Forms.IWin32Window** interface, to place the message box in front of. A message box is a modal dialog, which means no input (keyboard or mouse click) can occur except to objects on the modal form. The program must hide or close a modal form (typically in response to some user action) before input to another form can occur. The **System.Windows.Forms.IWin32Window** the message box will display of. The text to display in the message box. The text to display in the title bar of the message box. One of the **System.Windows.Forms.MessageBoxButtons** that specifies which buttons to display in the message box. One of the **System.Windows.Forms.MessageBoxIcon** values that specifies which icon to display in the message box. One fo the **System.Windows.Forms.MessageBoxDefaultButton** values which specifies which is the default button for the message box. One of the **System.Windows.Forms.MessageBoxOptions** values which specifies which display and association options will be used for the message box.

1        MessageBoxButtons enumeration (System.Windows.Forms)

2        *a)       ToString*

3  
4  
5        *Description*

6        Specifies constants defining which buttons to display on a  
7        **System.Windows.Forms.MessageBox** .

8        This enumeration is used by **System.Windows.Forms.MessageBox** .

9        *b)       ToString*

10        [C#]        public        const        MessageBoxButtons        AbortRetryIgnore;  
11        [C++]        public:        const        MessageBoxButtons        AbortRetryIgnore;  
12        [VB]        Public        Const        AbortRetryIgnore        As        MessageBoxButtons  
13        [JScript]        public        var        AbortRetryIgnore        :        MessageBoxButtons;

14  
15        *Description*

16        The message box contains **Abort** , **Retry** , and **Ignore** buttons.

17        *c)       ToString*

18  
19        [C#]        public        const        MessageBoxButtons        OK;  
20        [C++]        public:        const        MessageBoxButtons        OK;  
21        [VB]        Public        Const        OK        As        MessageBoxButtons  
22        [JScript]        public        var        OK        :        MessageBoxButtons;

23  
24        *Description*

The message box contains an **OK** button.

*d) ToString*

```
[C#]      public      const      MessageBoxButtons      OKCancel;
[C++]     public:     const      MessageBoxButtons      OKCancel;
[VB]      Public      Const      OKCancel      As      MessageBoxButtons
[JScript] public      var      OKCancel      :      MessageBoxButtons;
```

*Description*

The message box contains **OK** and **Cancel** buttons.

*e) ToString*

```
[C#]      public      const      MessageBoxButtons      RetryCancel;
[C++]     public:     const      MessageBoxButtons      RetryCancel;
[VB]      Public      Const      RetryCancel      As      MessageBoxButtons
[JScript] public      var      RetryCancel      :      MessageBoxButtons;
```

*Description*

The message box contains **Retry** and **Cancel** buttons.

*f) ToString*

```
[C#]      public      const      MessageBoxButtons      YesNo;
[C++]     public:     const      MessageBoxButtons      YesNo;
[VB]      Public      Const      YesNo      As      MessageBoxButtons
[JScript] public      var      YesNo      :      MessageBoxButtons;
```

*Description*

The message box contains **Yes** and **No** buttons.

*g) ToString*

```
[C#]      public      const      MessageBoxButtons      YesNoCancel;
[C++]     public:     const      MessageBoxButtons      YesNoCancel;
[VB]      Public      Const      YesNoCancel      As      MessageBoxButtons
[JScript] public      var      YesNoCancel      :      MessageBoxButtons;
```

*Description*

The message box contains **Yes** , **No** , and **Cancel** buttons.

MessageBoxDefaultButton enumeration (System.Windows.Forms)

*a) ToString*

*Description*

Specifies constants defining the default button on a **System.Windows.Forms.MessageBox** .

This enumeration is used by **System.Windows.Forms.MessageBox** .

*b) ToString*

```
[C#]      public      const      MessageBoxDefaultButton      Button1;
[C++]     public:     const      MessageBoxDefaultButton      Button1;
[VB]      Public      Const      Button1      As      MessageBoxDefaultButton
```

1 [JScript] public var Button1 : MessageBoxDefaultButton;

3 *Description*

4 The first button on the message box is the default button.

5 *c) ToString*

7 [C#] public const MessageBoxDefaultButton Button2;

8 [C++] public: const MessageBoxDefaultButton Button2;

9 [VB] Public Const Button2 As MessageBoxDefaultButton

10 [JScript] public var Button2 : MessageBoxDefaultButton;

12 *Description*

13 The second button on the message box is the default button.

14 *d) ToString*

16 [C#] public const MessageBoxDefaultButton Button3;

17 [C++] public: const MessageBoxDefaultButton Button3;

18 [VB] Public Const Button3 As MessageBoxDefaultButton

19 [JScript] public var Button3 : MessageBoxDefaultButton;

21 *Description*

22 The third button on the message box is the default button.

## MessageBoxIcon enumeration (System.Windows.Forms)

### a) *ToString*

#### *Description*

Specifies constants defining which information to display.

This enumeration is used by the **System.Windows.Forms.MessageBox** class. The description of each member of this enumeration contains a typical representation of the symbol. The actual graphic displayed is a function of the operating system constants. In current implementations there are four unique symbols with multiple values assigned to them.

### b) *ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Asterisk;         |
| [C++]     | public: | const | MessageBoxIcon | Asterisk;         |
| [VB]      | Public  | Const | Asterisk       | As MessageBoxIcon |
| [JScript] | public  | var   | Asterisk       | : MessageBoxIcon; |

#### *Description*

The message box contains a symbol consisting of a lowercase letter i in a circle.

### c) *ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Error;            |
| [C++]     | public: | const | MessageBoxIcon | Error;            |
| [VB]      | Public  | Const | Error          | As MessageBoxIcon |
| [JScript] | public  | var   | Error          | : MessageBoxIcon; |

*Description*

The message box contains a symbol consisting of white X in a circle with a red background.

*d) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Exclamation;      |
| [C++]     | public: | const | MessageBoxIcon | Exclamation;      |
| [VB]      | Public  | Const | Exclamation    | As MessageBoxIcon |
| [JScript] | public  | var   | Exclamation    | : MessageBoxIcon; |

*Description*

The message box contains a symbol consisting of an exclamation point in a triangle with a yellow background.

*e) ToString*

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Hand;             |
| [C++]     | public: | const | MessageBoxIcon | Hand;             |
| [VB]      | Public  | Const | Hand           | As MessageBoxIcon |
| [JScript] | public  | var   | Hand           | : MessageBoxIcon; |

*Description*

The message box contains a symbol consisting of a white X in a circle with a red background.



***f) ToString***

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Information;      |
| [C++]     | public: | const | MessageBoxIcon | Information;      |
| [VB]      | Public  | Const | Information    | As MessageBoxIcon |
| [JScript] | public  | var   | Information    | : MessageBoxIcon; |

***Description***

The message box contains a symbol consisting of a lowercase letter i in a circle.

***g) ToString***

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | None;             |
| [C++]     | public: | const | MessageBoxIcon | None;             |
| [VB]      | Public  | Const | None           | As MessageBoxIcon |
| [JScript] | public  | var   | None           | : MessageBoxIcon; |

***Description***

The message box contain no symbols.

***h) ToString***

|           |         |       |                |                   |
|-----------|---------|-------|----------------|-------------------|
| [C#]      | public  | const | MessageBoxIcon | Question;         |
| [C++]     | public: | const | MessageBoxIcon | Question;         |
| [VB]      | Public  | Const | Question       | As MessageBoxIcon |
| [JScript] | public  | var   | Question       | : MessageBoxIcon; |

### *Description*

The message box contains a symbol consisting of a question mark in a circle.

#### *i) ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | MessageBoxIcon | Stop;           |
| [C++]     | public: | const | MessageBoxIcon | Stop;           |
| [VB]      | Public  | Const | Stop As        | MessageBoxIcon  |
| [JScript] | public  | var   | Stop :         | MessageBoxIcon; |

### *Description*

The message box contains a symbol consisting of white X in a circle with a red background.

#### *j) ToString*

|           |         |       |                |                 |
|-----------|---------|-------|----------------|-----------------|
| [C#]      | public  | const | MessageBoxIcon | Warning;        |
| [C++]     | public: | const | MessageBoxIcon | Warning;        |
| [VB]      | Public  | Const | Warning As     | MessageBoxIcon  |
| [JScript] | public  | var   | Warning :      | MessageBoxIcon; |

### *Description*

The message box contains a symbol consisting of an exclamation-point in a triangle with a yellow background.

1        MessageBoxOptions enumeration (System.Windows.Forms)

2        *a)        ToString*

5        *Description*

6        Specifies options on a **System.Windows.Forms.MessageBox** .

7        This enumeration is used by **System.Windows.Forms.MessageBox** .

8        *b)        ToString*

10        [C#]        public        const        MessageBoxOptions        DefaultDesktopOnly;

11        [C++]        public:        const        MessageBoxOptions        DefaultDesktopOnly;

12        [VB]        Public        Const        DefaultDesktopOnly        As        MessageBoxOptions

13        [JScript]        public        var        DefaultDesktopOnly        :        MessageBoxOptions;

15        *Description*

16        The message box is displayed on the active desktop.

17        *c)        ToString*

19        [C#]        public        const        MessageBoxOptions        RightAlign;

20        [C++]        public:        const        MessageBoxOptions        RightAlign;

21        [VB]        Public        Const        RightAlign        As        MessageBoxOptions

22        [JScript]        public        var        RightAlign        :        MessageBoxOptions;

24        *Description*

25        The message box text is right-aligned.

d) *ToString*

```
[C#]      public      const      MessageBoxOptions      RtlReading;
[C++]     public:     const      MessageBoxOptions      RtlReading;
[VB]      Public      Const      RtlReading      As      MessageBoxOptions
[JScript] public      var      RtlReading      :      MessageBoxOptions;
```

*Description*

Specifies that the message box text is displayed with right to left reading order.

e) *ToString*

```
[C#]      public      const      MessageBoxOptions      ServiceNotification;
[C++]     public:     const      MessageBoxOptions      ServiceNotification;
[VB]      Public      Const      ServiceNotification      As      MessageBoxOptions
[JScript] public      var      ServiceNotification      :      MessageBoxOptions;
```

*Description*

The message box is displayed on the active desktop.

MethodInvoker delegate (System.Windows.Forms)

a) *ToString*

*Description*

Represents the method that handles the **Invoke** event for a method.

**System.Windows.Forms.MethodInvoker** provides a simple delegate that is used to invoke a method with a void parameter list. This delegate can be used

when making calls to a control's invoke method, or when you need a simple delegate but don't want to define one yourself.

MonthCalendar class (System.Windows.Forms)

a) *ToString*

*Description*

Represents a standard Windows month calendar control.

The **System.Windows.Forms.MonthCalendar** control allows the user to select a date and time using a visual display. You can limit the date and times that can be selected by setting the

**System.Windows.Forms.MonthCalendar.MinDate** and **System.Windows.Forms.MonthCalendar.MaxDate** properties.

b) *MonthCalendar*

*Example Syntax:*

c) *ToString*

|           |         |                           |
|-----------|---------|---------------------------|
| [C#]      | public  | MonthCalendar();          |
| [C++]     | public: | MonthCalendar();          |
| [VB]      | Public  | Sub New()                 |
| [JScript] | public  | function MonthCalendar(); |

*Description*

Initializes a new instance of the **System.Windows.Forms.MonthCalendar** class.

Creates a new **System.Windows.Forms.MonthCalendar** object.

- d) *AccessibilityObject*
- e) *AccessibleDefaultActionDescription*
- f) *AccessibleDescription*
- g) *AccessibleName*
- h) *AccessibleRole*
- i) *AllowDrop*
- j) *Anchor*
- k) *AnnuallyBoldedDates*
- l) *ToString*

#### *Description*

The array of **System.DateTime** objects that determines which annual days are shown in bold.

Using this property you can add multiple dates to the array of annually bolded dates. When you assign an array of dates the existing dates are cleared.

m) *BackColor*

n) *ToString*

```
[C#]      public      override      Color      BackColor      {get;      set;}
```

```
[C++] public: __property virtual Color get_BackColor();public: __property virtual  
void set_BackColor(Color);
```

```
[VB]      Overrides      Public      Property      BackColor      As      Color
```

```
[JScript] public function get BackColor() : Color;public function set  
BackColor(Color);
```

*Description*

*o) BackgroundImage*

*p) ToString*

```
[C#] public override Image BackgroundImage {get; set;}
```

```
[C++] public: __property virtual Image* get_BackgroundImage();public:
```

```
__property virtual void set_BackgroundImage(Image*);
```

```
[VB] Overrides Public Property BackgroundImage As Image
```

```
[JScript] public function get BackgroundImage() : Image;public function set  
BackgroundImage(Image);
```

*Description*

*q) BindingContext*

*r) BoldedDates*

*s) ToString*

*Description*

Gets or sets the array of **System.DateTime** objects that determines which nonrecurring dates are displayed in bold.

Using this property you can assign an array of bolded dates. When you assign an array of dates the existing dates are first cleared.

- t) *Bottom*
- u) *Bounds*
- v) *CalendarDimensions*
- w) *ToString*

*Description*

Gets or sets the number of columns and rows of months displayed.

Only one calendar year is displayed at a time, and the maximum number of months that can be displayed is 12. Valid combinations of columns and rows make a maximum product of 12; for values greater than 12 the display will be modified on a best-fit basis.



- x) *CanFocus*
- y) *CanSelect*
- z) *Capture*
- aa) *CausesValidation*
- bb) *ClientRectangle*
- cc) *ClientSize*
- dd) *CompanyName*
- ee) *Container*
- ff) *ContainsFocus*
- gg) *ContextMenu*
- hh) *Controls*
- ii) *Created*
- jj) *CreateParams*
- kk) *ToString*

#### *Description*

This is called when creating a window. Inheriting classes can override this to add extra functionality, but should not forget to first call `base.CreateParams()` to make sure the control continues to work correctly.

1 *ll) Cursor*

2 *mm) DataBindings*

3 *nn) DefaultImeMode*

4 *oo) ToString*

5  
6  
7 *Description*

8 Gets a value indicating the input method editor for the  
9 **System.Windows.Forms.MonthCalendar** .

10 *pp) DefaultSize*

11 *qq) ToString*

12  
13 [C#]       protected       override       Size       DefaultSize       {get;}

14 [C++]     protected:     \_\_property     virtual     Size     get\_DefaultSize();

15 [VB]     Overrides     Protected     ReadOnly     Property     DefaultSize     As     Size

16 [JScript]     protected     function     get     DefaultSize()     :     Size;

17  
18 *Description*

19 Gets the default size of the calendar.

20 The **System.Windows.Forms.MonthCalendar.DefaultSize** property includes  
21 the area necessary to include the "Today:" date display at the bottom of the  
22 calendar.  
23  
24  
25

- rr) *DesignMode*
- ss) *DisplayRectangle*
- tt) *Disposing*
- uu) *Dock*
- vv) *Enabled*
- ww) *Events*
- xx) *FirstDayOfWeek*
- yy) *ToString*

#### *Description*

Gets or sets the first day of the week as displayed in the month calendar.

The following example uses the **System.Windows.Forms.MonthCalendar.FirstDayOfWeek** to modify the calendar display to begin with a day of the week other than the default. This example assumes that a **System.Windows.Forms.MonthCalendar** control, named **monthCalendar1**, and a **System.Windows.Forms.ComboBox** control, has been added to a **System.Windows.Forms.Form** and that this method is placed within the form and called from it.

- zz) *Focused*
- aaa) *Font*
- bbb) *FontHeight*
- ccc) *ForeColor*
- ddd) *ToString*

#### *Description*

1  
2        *eee) Handle*

3        *fff) HasChildren*

4        *ggg) Height*

5        *hhh) ImeMode*

6        *iii) ToString*

7  
8  
9        *Description*

10       Gets or sets the Input Method Editor(IME) mode supported by this control.

11       *jjj) InvokeRequired*

12       *kkk) IsAccessible*

13       *lll) IsDisposed*

14       *mmm) IsHandleCreated*

15       *nnn) Left*

16       *ooo) Location*

17       *ppp) MaxDate*

18       *qqq) ToString*

19  
20  
21  
22       *Description*

23       Gets or sets the maximum allowable date.

1            **rrr)    *MaxSelectionCount***

2            **sss)    *ToString***

3  
4 [C#]        public        int        MaxSelectionCount        {get;        set;}

5 [C++] public: \_\_property int get\_MaxSelectionCount();public: \_\_property void  
6 set\_MaxSelectionCount(int);

7 [VB]        Public        Property        MaxSelectionCount        As        Integer

8 [JScript] public function get MaxSelectionCount() : int;public function set  
9 MaxSelectionCount(int);

10  
11 *Description*

12 The maximum number of days that can be selected in a month calendar control.

13 Setting this property does not affect the current selection range.

14            **ttt)    *MinDate***

15            **uuu)    *ToString***

16  
17 [C#]        public        DateTime        MinDate        {get;        set;}

18 [C++] public: \_\_property DateTime get\_MinDate();public: \_\_property void  
19 set\_MinDate(DateTime);

20 [VB]        Public        Property        MinDate        As        DateTime

21 [JScript] public function get MinDate() : DateTime;public function set  
22 MinDate(DateTime);

23  
24 *Description*

25 Gets or sets the minimum allowable date.

rvv) *MonthlyBoldedDates*

www) *ToString*

```
[C#]    public    DateTime[]    MonthlyBoldedDates    {get;    set;}
[C++]   public:   __property    DateTime    get_MonthlyBoldedDates();public:
__property          void          set_MonthlyBoldedDates(DateTime[]);
[VB]    Public    Property    MonthlyBoldedDates    As    DateTime    ()
[JavaScript] public function get MonthlyBoldedDates() : DateTime[];public function
set          MonthlyBoldedDates(DateTime[]);
```

### *Description*

The array of **System.DateTime** objects that determine which monthly days to bold.

Using this property you assign an array of monthly bolded dates. When you assign an array of dates, any preexisting dates are cleared.

xxx) *Name*

yyy) *Parent*

zzz) *ProductName*

aaaa) *ProductVersion*

bbbb) *RecreatingHandle*

cccc) *Region*

dddd) *RenderRightToLeft*

eeee) *ResizeRedraw*

ffff) *Right*

gggg) *RightToLeft*

hhhh) *ScrollChange*

iiii) *ToString*

### *Description*

The scroll rate for a month calendar control.

The scroll rate is the number of months that the control moves its display when the user clicks a scroll button. If this value is zero, the number of months is reset to the default, which is the number of months displayed in the control. The maximum value is 20000.

jjjj) *SelectionEnd*

kkkk) *ToString*

[C#]            public            DateTime            SelectionEnd            {get;            set;}

[C++] public: \_\_property DateTime get\_SelectionEnd();public: \_\_property void

1 set\_SelectionEnd(DateTime);

2 [VB] Public Property SelectionEnd As DateTime

3 [JScript] public function get SelectionEnd() : DateTime;public function set

4 SelectionEnd(DateTime);

6 *Description*

7 Gets or sets the end date of the selected range of dates.

8 If you set **System.Windows.Forms.MonthCalendar.SelectionEnd** to a date  
that is earlier than the current

9 **System.Windows.Forms.MonthCalendar.SelectionStart** , then

10 **System.Windows.Forms.MonthCalendar.SelectionStart** is automatically  
set equal to **System.Windows.Forms.MonthCalendar.SelectionEnd** .

11 *III) SelectionRange*

12 *mmm)ToString*

14 [C#] public SelectionRange SelectionRange {get; set;}

15 [C++] public: \_\_property SelectionRange\* get\_SelectionRange();public:

16 \_\_property void set\_SelectionRange(SelectionRange\*);

17 [VB] Public Property SelectionRange As SelectionRange

18 [JScript] public function get SelectionRange() : SelectionRange;public function set

19 SelectionRange(SelectionRange);

21 *Description*

22 Retrieves the selected range of dates for a month calendar control.

23 Setting this property is functionally equivalent to using the

24 **System.Windows.Forms.MonthCalendar.SetSelectionRange(System.Da  
teTime,System.DateTime)** method. You can set the start and end dates  
separately by setting either



**System.Windows.Forms.MonthCalendar.SelectionStart** or  
**System.Windows.Forms.MonthCalendar.SelectionEnd** .

*nnnn) SelectionStart*

*oooo) ToString*

[C#]        public        DateTime        SelectionStart        {get;        set;}

[C++] public: \_\_property DateTime get\_SelectionStart();public: \_\_property void  
set\_SelectionStart(DateTime);

[VB]        Public        Property        SelectionStart        As        DateTime

[JScript] public function get SelectionStart() : DateTime;public function set  
SelectionStart(DateTime);

*Description*

Gets or sets the start date of the selected range of dates.

If you set **System.Windows.Forms.MonthCalendar.SelectionStart** to a date that is later than the current **System.Windows.Forms.MonthCalendar.SelectionEnd** , then **System.Windows.Forms.MonthCalendar.SelectionEnd** is automatically set equal to **System.Windows.Forms.MonthCalendar.SelectionStart** .

*pppp) ShowFocusCues*

*qqqq) ShowKeyboardCues*

*rrrr) ShowToday*

*ssss) ToString*

*Description*

Gets or sets a value indicating whether the date represented by the **System.Windows.Forms.MonthCalendar.TodayDate** property is shown at the bottom of the control.

The date is shown in the format specified by the system settings for the short date format.

*tttt) ShowTodayCircle*

*uuuu) ToString*

[C#]        public        bool        ShowTodayCircle        {get;        set;}

[C++] public: \_\_property bool get\_ShowTodayCircle();public: \_\_property void set\_ShowTodayCircle(bool);

[VB]        Public        Property        ShowTodayCircle        As        Boolean

[JScript] public function get ShowTodayCircle() : Boolean;public function set ShowTodayCircle(Boolean);

#### *Description*

Gets or sets a value indicating whether today's date is circled.

*vvvv) ShowWeekNumbers*

*www)ToString*

[C#]        public        bool        ShowWeekNumbers        {get;        set;}

[C++] public: \_\_property bool get\_ShowWeekNumbers();public: \_\_property void set\_ShowWeekNumbers(bool);

[VB]        Public        Property        ShowWeekNumbers        As        Boolean

[JScript] public function get ShowWeekNumbers() : Boolean;public function set ShowWeekNumbers(Boolean);

*Description*

Gets or sets a value indicating whether the month calendar control displays week numbers (1-52) to the left of each row of days.

*xxxx) SingleMonthSize*

*yyyy) ToString*

|           |         |            |                 |                           |
|-----------|---------|------------|-----------------|---------------------------|
| [C#]      | public  | Size       | SingleMonthSize | {get;}                    |
| [C++]     | public: | __property | Size            | get_SingleMonthSize();    |
| [VB]      | Public  | ReadOnly   | Property        | SingleMonthSize As Size   |
| [JScript] | public  | function   | get             | SingleMonthSize() : Size; |

*Description*

Gets the minimum size to display one month of the calendar.

The size information is presented in the form of x and y members representing the minimum width and height required to display one month in the control. The minimum required window size for a month calendar control depends on the currently selected font.

zzzz) *Site*

aaaaa) *Size*

bbbbbb) *TabIndex*

cccc) *TabStop*

dddd) *Tag*

eeee) *Text*

fffff) *ToString*

## *Description*

ggggg) *TitleBackColor*

hhhhh) *ToString*

[C#]        public        Color        TitleBackColor        {get;        set;}

[C++] public: \_\_property Color get\_TitleBackColor();public: \_\_property void  
set\_TitleBackColor(Color);

[VB]        Public        Property        TitleBackColor        As        Color

[JScript] public function get TitleBackColor() : Color;public function set  
TitleBackColor(Color);

## *Description*

Gets or sets a value indicating the background color of the calendar's title area.

The font color of the days of the week text depends on the  
**System.Windows.Forms.MonthCalendar.TitleBackColor** property. Setting

the **System.Windows.Forms.MonthCalendar.TitleBackColor** equal to the **System.Windows.Forms.Control.BackColor** for the main display area of the calendar causes the day of the week text to become unreadable.

*iiii) TitleForeColor*

*jjjj) ToString*

[C#]        public        Color        TitleForeColor        {get;        set;}

[C++] public: \_\_property Color get\_TitleForeColor();public: \_\_property void  
set\_TitleForeColor(Color);

[VB]        Public        Property        TitleForeColor        As        Color

[JScript] public function get TitleForeColor() : Color;public function set  
TitleForeColor(Color);

### *Description*

Gets or sets a value indicating the foreground color of the calendar's title area.

*kkkkk) TodayDate*

*llll) ToString*

[C#]        public        DateTime        TodayDate        {get;        set;}

[C++] public: \_\_property DateTime get\_TodayDate();public: \_\_property void  
set\_TodayDate(DateTime);

[VB]        Public        Property        TodayDate        As        DateTime

[JScript] public function get TodayDate() : DateTime;public function set  
TodayDate(DateTime);

1  
2 *Description*

3 Gets or sets the value that is used by  
4 **System.Windows.Forms.MonthCalendar** as today's date.

5 By default the **System.Windows.Forms.MonthCalendar.TodayDate**  
6 property returns the current system time, and the  
7 **System.Windows.Forms.MonthCalendar.TodayDateSet** property is **false** .  
8 Setting the **System.Windows.Forms.MonthCalendar.TodayDate** property  
9 sets the **System.Windows.Forms.MonthCalendar.TodayDateSet** property  
10 to **true** , and from that point the value returned by the  
11 **System.Windows.Forms.MonthCalendar.TodayDate** property is the one  
12 the user sets.

13 *mmmmm)TodayDateSet*

14 *nnnnn)ToString*

15 [C#]            public            bool            TodayDateSet            {get;}  
16 [C++]           public:           \_\_property           bool           get\_TodayDateSet();  
17 [VB]    Public    ReadOnly    Property    TodayDateSet    As    Boolean  
18 [JScript]    public    function    get    TodayDateSet()    :    Boolean;

19 *Description*

20 Gets a value indicating whether the  
21 **System.Windows.Forms.MonthCalendar.TodayDate** property has been  
22 explicitly set.  
23  
24  
25

ooooo) *Top*

ppppp) *TopLevelControl*

qqqqq) *TrailingForeColor*

rrrrr) *ToString*

#### *Description*

Gets or sets a value indicating the color of days in months that are not fully displayed in the control.

When the calendar is displayed, there can be dates that precede and follow the months that are fully displayed. Using the **System.Windows.Forms.MonthCalendar.TrailingForeColor** property; you can modify the color of the text in which those days are displayed.

sssss) *Visible*

ttttt) *Width*

uuuuu) *WindowTarget*

vvvvv) *ToString*

wwwww) *ToString*

#### *Description*

Occurs when the date selected in the **System.Windows.Forms.MonthCalendar** changes.

When you create a **System.Windows.Forms.MonthCalendar.DateChanged** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

### xxxxx) ToString

```
1
2 [C#]      public      event      DateRangeEventHandler      DateSelected;
3 [C++]     public:     __event    DateRangeEventHandler*      DateSelected;
4 [VB]      Public      Event      DateSelected      As      DateRangeEventHandler
5
6
```

### Description

Occurs when a date is selected in the month calendar.

When you create a **System.Windows.Forms.MonthCalendar.DateSelected** delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

### yyyyy) ToString

### zzzzz) AddAnnuallyBoldedDate

```
14
15 [C#]      public      void      AddAnnuallyBoldedDate(DateTime      date);
16 [C++]     public:     void      AddAnnuallyBoldedDate(DateTime      date);
17 [VB]      Public      Sub      AddAnnuallyBoldedDate(ByVal      date      As      DateTime)
18 [JScript] public      function  AddAnnuallyBoldedDate(date      :      DateTime);
19
```

### Description

Adds a day that is displayed in bold on an annual basis in the month calendar.

You must call

**System.Windows.Forms.MonthCalendar.UpdateBoldedDates** afterwards to update the display. The date to be displayed in bold.



### *aaaaaa)AddBoldedDate*

```
1
2 [C#]      public      void      AddBoldedDate(DateTime      date);
3
4 [C++]      public:      void      AddBoldedDate(DateTime      date);
5
6 [VB]      Public      Sub      AddBoldedDate(ByVal      date      As      DateTime)
7
8 [JScript]      public      function      AddBoldedDate(date      :      DateTime);
9
```

#### *Description*

Adds a day to be displayed in bold in the month calendar.

You must call

**System.Windows.Forms.MonthCalendar.UpdateBoldedDates** afterwards to update the display. The date to be displayed in bold.

### *bbbbbb)AddMonthlyBoldedDate*

```
12
13 [C#]      public      void      AddMonthlyBoldedDate(DateTime      date);
14
15 [C++]      public:      void      AddMonthlyBoldedDate(DateTime      date);
16
17 [VB]      Public      Sub      AddMonthlyBoldedDate(ByVal      date      As      DateTime)
18
19 [JScript]      public      function      AddMonthlyBoldedDate(date      :      DateTime);
20
```

#### *Description*

Adds a day that is displayed in bold on a monthly basis in the month calendar.

You must call

**System.Windows.Forms.MonthCalendar.UpdateBoldedDates** afterwards to update the display. The date to be displayed in bold.

### *cccccc)CreateHandle*

```
23
24
25 [C#]      protected      override      void      CreateHandle();
```

|           |            |                   |                 |
|-----------|------------|-------------------|-----------------|
| [C++]     | protected: | void              | CreateHandle(); |
| [VB]      | Overrides  | Protected Sub     | CreateHandle()  |
| [JScript] | protected  | override function | CreateHandle(); |

#### Description

**dddddd)Dispose**

|           |            |               |                                     |                               |
|-----------|------------|---------------|-------------------------------------|-------------------------------|
| [C#]      | protected  | override      | void                                | Dispose(bool disposing);      |
| [C++]     | protected: | void          | Dispose(bool disposing);            |                               |
| [VB]      | Overrides  | Protected Sub | Dispose(ByVal disposing As Boolean) |                               |
| [JScript] | protected  | override      | function                            | Dispose(disposing : Boolean); |

#### Description

Called to cleanup a MonthCalendar. Normally you do not need to call this as the garbage collector will cleanup the buffer for you. However, there may be times when you may want to expedite the garbage collectors cleanup.

**eeeeee)GetDisplayRange**

|           |                 |                                           |                                                      |
|-----------|-----------------|-------------------------------------------|------------------------------------------------------|
| [C#]      | public          | SelectionRange                            | GetDisplayRange(bool visible);                       |
| [C++]     | public:         | SelectionRange*                           | GetDisplayRange(bool visible);                       |
| [VB]      | Public Function | GetDisplayRange(ByVal visible As Boolean) | As SelectionRange                                    |
| [JScript] | public          | function                                  | GetDisplayRange(visible : Boolean) : SelectionRange; |

#### Description

Retrieves date information that represents the low and high limits of the control's displayed dates.

*Return Value:* The begin and end dates of the displayed calendar.

The

**System.Windows.Forms.MonthCalendar.GetDisplayRange(System.Boolean)** method returns a

**System.Windows.Forms.MonthCalendar.SelectionRange** object that contains the begin and end dates that are displayed in the current view of the control. You can modify the returned range based on the value you specify for the *visible* parameter. By specifying **false** you can retrieve all dates displayed on the control; specifying **true** retrieves only those dates that are part of fully displayed months. **true** to retrieve only the dates that are fully contained in displayed months; otherwise, **false**.

*fffff) HitTest*

[C#]            public            HitTestInfo            HitTest(Point            point);

[C++]            public:            HitTestInfo\*            HitTest(Point            point);

[VB]   Public   Function   HitTest(ByVal   point   As   Point)   As   HitTestInfo

[JScript]   public   function   HitTest(point   :   Point)   :   HitTestInfo;

### *Description*

Returns an object with information on which portion of a month calendar control is at a location specified by a **System.Drawing.Point** object.

*Return Value:* A **System.Windows.Forms.MonthCalendar.HitTestInfo** object that contains information about the specified point on the **System.Windows.Forms.MonthCalendar**. A **System.Drawing.Point** object containing the x and y coordinates of the point to be hit-tested.

*gggggg)HitTest*

[C#]            public            HitTestInfo            HitTest(int            x,            int            y);

[C++]            public:            HitTestInfo\*            HitTest(int            x,            int            y);

[VB]   Public   Function   HitTest(ByVal   x   As   Integer, ByVal   y   As   Integer)   As

## HitTestInfo

[JScript] public function HitTest(x : int, y : int) : HitTestInfo; Determines which element of the calendar is at a specified location.

### *Description*

Returns a **System.Windows.Forms.MonthCalendar.HitTestInfo** object with information on which portion of a month calendar control is at a specified x and y location.

*Return Value:* A **System.Windows.Forms.MonthCalendar.HitTestInfo** object that contains information about the specified point on the **System.Windows.Forms.MonthCalendar**. The x coordinate of the point to be hit-tested. The y coordinate of the point to be hit-tested.

### *hhhhh)IsInputKey*

[C#] protected override bool IsInputKey(Keys keyData);

[C++] protected: bool IsInputKey(Keys keyData);

[VB] Overrides Protected Function IsInputKey(ByVal keyData As Keys) As Boolean

[JScript] protected override function IsInputKey(keyData : Keys) : Boolean;

### *Description*

Handling special input keys, such as pgup, pgdown, home, end, etc...

### *iiiiii) OnBackColorChanged*

[C#] protected override void OnBackColorChanged(EventArgs e);

[C++] protected: void OnBackColorChanged(EventArgs\* e);

[VB] Overrides Protected Sub OnBackColorChanged(ByVal e As EventArgs)

1 [JScript] protected override function OnBackColorChanged(e : EventArgs);

2  
3 *Description*

4  
5 *jjjjj) OnDateChanged*

6  
7 [C#] protected virtual void OnDateChanged(DateRangeEventArgs drevent);

8 [C++] protected: virtual void OnDateChanged(DateRangeEventArgs\* drevent);

9 [VB] Overridable Protected Sub OnDateChanged(ByVal drevent As  
10 DateRangeEventArgs)

11 [JScript] protected function OnDateChanged(drevent : DateRangeEventArgs);

12  
13 *Description*

14 Raises the **System.Windows.Forms.MonthCalendar.DateChanged** event.

15 Raising an event invokes the event handler through a delegate. For more  
16 information, see . A **System.Windows.Forms.DateRangeEventArgs** that  
contains the event data.

17 *kkkkkk)OnDateSelected*

18  
19 [C#] protected virtual void OnDateSelected(DateRangeEventArgs drevent);

20 [C++] protected: virtual void OnDateSelected(DateRangeEventArgs\* drevent);

21 [VB] Overridable Protected Sub OnDateSelected(ByVal drevent As  
22 DateRangeEventArgs)

23 [JScript] protected function OnDateSelected(drevent : DateRangeEventArgs);

24  
25 *Description*

Raises the **System.Windows.Forms.MonthCalendar.DateSelected** event.

Raising an event invokes the event handler through a delegate. For more information, see . A **System.Windows.Forms.DateRangeEventArgs** that contains the event data.

#### *IIIIII) OnFontChanged*

[C#]      protected      override      void      OnFontChanged(EventArgs      e);

[C++]      protected:      void      OnFontChanged(EventArgs\*      e);

[VB] Overrides Protected Sub OnFontChanged(ByVal e As EventArgs)

[JScript] protected override function OnFontChanged(e : EventArgs);

#### *Description*

Raises the **System.Windows.Forms.Control.FontChanged** event.

The **System.Windows.Forms.MonthCalendar.OnFontChanged(System.EventArgs)** method resizes the control based on the new font size. This can cause the calendar to overlap other controls on the form if the potential sizing issues have not been taken into account. An **System.EventArgs** that contains the event data.

#### *mmmmmm)OnForeColorChanged*

[C#]      protected      override      void      OnForeColorChanged(EventArgs      e);

[C++]      protected:      void      OnForeColorChanged(EventArgs\*      e);

[VB] Overrides Protected Sub OnForeColorChanged(ByVal e As EventArgs)

[JScript] protected override function OnForeColorChanged(e : EventArgs);

#### *Description*

### *nnnnnn)OnHandleCreated*

```
[C#]    protected    override    void    OnHandleCreated(EventArgs    e);
[C++]    protected:    void    OnHandleCreated(EventArgs*    e);
[VB]    Overrides Protected Sub OnHandleCreated(ByVal e As EventArgs)
[JScript] protected override function OnHandleCreated(e : EventArgs);
```

#### *Description*

Overrides Control.OnHandleCreated() Overrides Control.OnHandleCreated()

### *oooooo)RemoveAllAnnuallyBoldedDates*

```
[C#]    public    void    RemoveAllAnnuallyBoldedDates();
[C++]    public:    void    RemoveAllAnnuallyBoldedDates();
[VB]    Public    Sub    RemoveAllAnnuallyBoldedDates()
[JScript]    public    function    RemoveAllAnnuallyBoldedDates();
```

#### *Description*

Removes all the annually bolded dates.

This method clears all dates from the **System.Windows.Forms.MonthCalendar.AnnuallyBoldedDates** array. To remove a single date from the bolded dates, use the **System.Windows.Forms.MonthCalendar.RemoveAnnuallyBoldedDate(System.DateTime)** method.

### *pppppp)RemoveAllBoldedDates*

```
[C#]    public    void    RemoveAllBoldedDates();
[C++]    public:    void    RemoveAllBoldedDates();
```

1 [VB] Public Sub RemoveAllBoldedDates()

2 [JScript] public function RemoveAllBoldedDates();

3  
4 *Description*

5 Removes all the nonrecurring bolded dates.

6 This method clears the  
7 **System.Windows.Forms.MonthCalendar.BoldedDates** array. To remove a  
8 single date from the bolded dates, use the  
9 **System.Windows.Forms.MonthCalendar.RemoveBoldedDate(System.D**  
10 **ateTime)** method.

11 *qqqqqq)RemoveAllMonthlyBoldedDates*

12 [C#] public void RemoveAllMonthlyBoldedDates();

13 [C++] public: void RemoveAllMonthlyBoldedDates();

14 [VB] Public Sub RemoveAllMonthlyBoldedDates()

15 [JScript] public function RemoveAllMonthlyBoldedDates();

16 *Description*

17 Removes all the monthly bolded dates.

18 This method clears the  
19 **System.Windows.Forms.MonthCalendar.MonthlyBoldedDates** array. To  
20 remove a single date from the bolded dates, use the  
21 **System.Windows.Forms.MonthCalendar.RemoveMonthlyBoldedDate(Sy**  
22 **stem.DateTime)** method.

23 *rrrrrr) RemoveAnnuallyBoldedDate*

24 [C#] public void RemoveAnnuallyBoldedDate(DateTime date);

25 [C++] public: void RemoveAnnuallyBoldedDate(DateTime date);



1 [VB] Public Sub RemoveAnnuallyBoldedDate(ByVal date As DateTime)

2 [JScript] public function RemoveAnnuallyBoldedDate(date : DateTime);

4 *Description*

5 Removes the specified date from the list of annually bolded dates.

6 If the specified date occurs more than once in the date list, then only the first  
7 date is removed. When comparing dates, only the day and month are used. You  
8 must call **System.Windows.Forms.MonthCalendar.UpdateBoldedDates** to  
ensure that the display is updated to reflect the removal. The date to remove  
from the list.

9 *sssss) RemoveBoldedDate*

11 [C#] public void RemoveBoldedDate(DateTime date);

12 [C++] public: void RemoveBoldedDate(DateTime date);

13 [VB] Public Sub RemoveBoldedDate(ByVal date As DateTime)

14 [JScript] public function RemoveBoldedDate(date : DateTime);

16 *Description*

17 Removes the specified date from the list of nonrecurring bolded dates.

18 If the specified date occurs more than once in the date list, then only the first  
19 date is removed. You must call  
20 **System.Windows.Forms.MonthCalendar.UpdateBoldedDates** to ensure  
that the display is updated to reflect the removal. The date to remove from the  
list.

21 *ttttt) RemoveMonthlyBoldedDate*

23 [C#] public void RemoveMonthlyBoldedDate(DateTime date);

24 [C++] public: void RemoveMonthlyBoldedDate(DateTime date);

1 [VB] Public Sub RemoveMonthlyBoldedDate(ByVal date As DateTime)

2 [JScript] public function RemoveMonthlyBoldedDate(date : DateTime);

4 *Description*

5 Removes the specified date from the list of monthly bolded dates.

6 If the specified date occurs more than once in the date list, then only the first  
7 date is removed. When comparing dates, only the day and month are used. You  
8 must call **System.Windows.Forms.MonthCalendar.UpdateBoldedDates** to  
ensure that the display is updated to reflect the removal. The date to remove  
from the list.

9 *uuuuuu)SetBoundsCore*

11 [C#] protected override void SetBoundsCore(int x, int y, int width, int height,  
12 BoundsSpecified specified);

13 [C++] protected: void SetBoundsCore(int x, int y, int width, int height,  
14 BoundsSpecified specified);

15 [VB] Overrides Protected Sub SetBoundsCore(ByVal x As Integer, ByVal y As  
16 Integer, ByVal width As Integer, ByVal height As Integer, ByVal specified As  
17 BoundsSpecified)

18 [JScript] protected override function SetBoundsCore(x : int, y : int, width : int,  
19 height : int, specified : BoundsSpecified); Overrides Control.SetBoundsCore to  
20 enforce auto-sizing.

21 *vvvvvv)SetCalendarDimensions*

23 [C#] public void SetCalendarDimensions(int x, int y);

24 [C++] public: void SetCalendarDimensions(int x, int y);

[VB] Public Sub SetCalendarDimensions(ByVal x As Integer, ByVal y As Integer)

[JScript] public function SetCalendarDimensions(x : int, y : int);

#### *Description*

Sets the number of columns and rows of months to display.

The maximum number of months that will be displayed is 12; restricted to one calendar year. The product of the *x* and *y* parameters should be 12 or less. For values of *x* and *y* that have a product of more than 12, the greater of the *x* and *y* values will be iteratively reduced until the product is 12 or less. The number of columns. The number of rows.

#### *wwwwww)SetDate*

[C#] public void SetDate(DateTime date);

[C++] public: void SetDate(DateTime date);

[VB] Public Sub SetDate(ByVal date As DateTime)

[JScript] public function SetDate(date : DateTime);

#### *Description*

Sets a date as the current selected date.

This method sets the **System.Windows.Forms.MonthCalendar.SelectionStart** and the **System.Windows.Forms.MonthCalendar.SelectionEnd** properties to the specified date and is the functional equivalent of setting the selection range to a single day through the **System.Windows.Forms.MonthCalendar.SetSelectionRange(System.DateTime, System.DateTime)** method or the **System.Windows.Forms.MonthCalendar.SelectionRange** property. The date to be selected.

### xxxxxx)SetSelectionRange

```
[C#] public void SetSelectionRange(DateTime date1, DateTime date2);
[C++] public: void SetSelectionRange(DateTime date1, DateTime date2);
[VB] Public Sub SetSelectionRange(ByVal date1 As DateTime, ByVal date2 As
DateTime)
[JScript] public function SetSelectionRange(date1 : DateTime, date2 : DateTime);
```

#### Description

Sets the selected dates in a month calendar control to the specified date range.

Using this method is functionally equivalent to setting the **System.Windows.Forms.MonthCalendar.SelectionRange** property. You can set the start and end dates separately by setting either **System.Windows.Forms.MonthCalendar.SelectionStart** or **System.Windows.Forms.MonthCalendar.SelectionEnd** . The beginning date of the selection range. The end date of the selection range.

### yyyyyy)ToString

```
[C#] public override string ToString();
[C++] public: String* ToString();
[VB] Overrides Public Function ToString() As String
[JScript] public override function ToString() : String;
```

#### Description

Returns a string representation for this control.

*Return Value:* String Returns a string representation for this control.

### zzzzzz) UpdateBoldedDates

|           |         |          |                      |
|-----------|---------|----------|----------------------|
| [C#]      | public  | void     | UpdateBoldedDates(); |
| [C++]     | public: | void     | UpdateBoldedDates(); |
| [VB]      | Public  | Sub      | UpdateBoldedDates()  |
| [JScript] | public  | function | UpdateBoldedDates(); |

#### Description

Repaints the bolded dates reflect the dates set in the lists of bolded dates.

Use the **System.Windows.Forms.MonthCalendar.UpdateBoldedDates** method to reflect changes made to **System.Windows.Forms.MonthCalendar.AnnuallyBoldedDates** , **System.Windows.Forms.MonthCalendar.MonthlyBoldedDates** , or **System.Windows.Forms.MonthCalendar.BoldedDates** either directly by modifying elements of the array or through any of the add or remove methods provided to modify the lists.

### aaaaaaa) WndProc

|           |            |           |          |                             |
|-----------|------------|-----------|----------|-----------------------------|
| [C#]      | protected  | override  | void     | WndProc(ref Message m);     |
| [C++]     | protected: |           | void     | WndProc(Message* m);        |
| [VB]      | Overrides  | Protected | Sub      | WndProc(ByRef m As Message) |
| [JScript] | protected  | override  | function | WndProc(m : Message);       |

#### Description

Overridden wndProc Overridden wndProc

## MouseButtons enumeration (System.Windows.Forms)

### a) *WndProc*

#### *Description*

Specifies constants that define which mouse button was pressed.

This enumeration is used by many classes, including **System.Windows.Forms.AxHost** , **System.Windows.Forms.Control** , **System.Windows.Forms.DataGrid** , **System.Windows.Forms.Form** , **System.Windows.Forms.RadioButton** , **System.Windows.Forms.Splitter** , **System.Windows.Forms.StatusBar** , and **System.Windows.Forms.UpDownBase** .

### b) *WndProc*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | MouseButtons | Left;           |
| [C++]     | public: | const | MouseButtons | Left;           |
| [VB]      | Public  | Const | Left         | As MouseButtons |
| [JScript] | public  | var   | Left         | : MouseButtons; |

#### *Description*

The left mouse button was pressed.

### c) *WndProc*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | MouseButtons | Middle;         |
| [C++]     | public: | const | MouseButtons | Middle;         |
| [VB]      | Public  | Const | Middle       | As MouseButtons |
| [JScript] | public  | var   | Middle       | : MouseButtons; |

*Description*

The middle mouse button was pressed.

*d) WndProc*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | MouseButtons | None;           |
| [C++]     | public: | const | MouseButtons | None;           |
| [VB]      | Public  | Const | None         | As MouseButtons |
| [JScript] | public  | var   | None         | : MouseButtons; |

*Description*

No mouse button was pressed.

*e) WndProc*

|           |         |       |              |                 |
|-----------|---------|-------|--------------|-----------------|
| [C#]      | public  | const | MouseButtons | Right;          |
| [C++]     | public: | const | MouseButtons | Right;          |
| [VB]      | Public  | Const | Right        | As MouseButtons |
| [JScript] | public  | var   | Right        | : MouseButtons; |

*Description*

The right mouse button was pressed.

*f) WndProc*

|       |         |       |              |           |
|-------|---------|-------|--------------|-----------|
| [C#]  | public  | const | MouseButtons | XButton1; |
| [C++] | public: | const | MouseButtons | XButton1; |

|           |        |       |          |    |               |
|-----------|--------|-------|----------|----|---------------|
| [VB]      | Public | Const | XButton1 | As | MouseButtons  |
| [JScript] | public | var   | XButton1 | :  | MouseButtons; |

*Description*

The first XButton was pressed.

*g) WndProc*

|           |         |       |              |           |               |
|-----------|---------|-------|--------------|-----------|---------------|
| [C#]      | public  | const | MouseButtons | XButton2; |               |
| [C++]     | public: | const | MouseButtons | XButton2; |               |
| [VB]      | Public  | Const | XButton2     | As        | MouseButtons  |
| [JScript] | public  | var   | XButton2     | :         | MouseButtons; |

*Description*

The second XButton was pressed.

MouseEventArgs class (System.Windows.Forms)

*a) ToString*

*Description*

Provides data for the **MouseUp** , **MouseDown** , and **MouseMove** events.

The **MouseDown** event occurs when the user presses the mouse button while the pointer is over a control. The **MouseUp** event occurs when the user releases the mouse button while the pointer remains over the control. The **MouseMove** event occurs when the user moves the mouse pointer over a control. A **MouseEventArgs** specifies which mouse button is pressed, how many times, the coordinates of the mouse click, and the amount the mouse wheel moved.



b) *MouseEventArgs*

*Example Syntax:*

c) *ToString*

[C#] public MouseEventArgs(MouseButtons button, int clicks, int x, int y, int delta);

[C++] public: MouseEventArgs(MouseButtons button, int clicks, int x, int y, int delta);

[VB] Public Sub New(ByVal button As MouseButtons, ByVal clicks As Integer, ByVal x As Integer, ByVal y As Integer, ByVal delta As Integer)

[JScript] public function MouseEventArgs(button : MouseButtons, clicks : int, x : int, y : int, delta : int);

*Description*

Initializes a new instance of the **System.Windows.Forms.MouseEventArgs** class. One of the **System.Windows.Forms.MouseButtons** values indicating which mouse button was pressed. The number of times a mouse button was pressed. The x-coordinate of a mouse click, in pixels. The y-coordinate of a mouse click, in pixels. A signed count of the number of detents the wheel has rotated.

d) *Button*

e) *ToString*

[C#] public MouseButtons Button {get;}

[C++] public: \_\_property MouseButtons get\_Button();

[VB] Public ReadOnly Property Button As MouseButtons

[JScript] public function get Button() : MouseButtons;

*Description*

Gets which mouse button was pressed.

*f) Clicks*

*g) ToString*

|           |         |            |              |                   |
|-----------|---------|------------|--------------|-------------------|
| [C#]      | public  | int        | Clicks       | {get;}            |
| [C++]     | public: | __property | int          | get_Clicks();     |
| [VB]      | Public  | ReadOnly   | Property     | Clicks As Integer |
| [JScript] | public  | function   | get Clicks() | : int;            |

*Description*

Gets the number of times the mouse button was pressed and released.

*h) Delta*

*i) ToString*

|           |         |            |             |                  |
|-----------|---------|------------|-------------|------------------|
| [C#]      | public  | int        | Delta       | {get;}           |
| [C++]     | public: | __property | int         | get_Delta();     |
| [VB]      | Public  | ReadOnly   | Property    | Delta As Integer |
| [JScript] | public  | function   | get Delta() | : int;           |

*Description*

Gets a signed count of the number of detents the mouse wheel has rotated. A detent is the rotation of the mouse wheel one notch.

The mouse wheel combines the features of a wheel and a mouse button. The wheel has discrete, evenly-spaced notches. When you rotate the wheel, a wheel

message is sent as each notch is encountered. One wheel notch, a detent, is defined by the windows constant WHEEL\_DELTA, which is 120. A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user.

j) *X*

k) *ToString*

|           |         |            |          |              |
|-----------|---------|------------|----------|--------------|
| [C#]      | public  | int        | X        | {get;}       |
| [C++]     | public: | __property | int      | get_X();     |
| [VB]      | Public  | ReadOnly   | Property | X As Integer |
| [JScript] | public  | function   | get X()  | : int;       |

#### *Description*

Gets the x-coordinate of a mouse click.

The mouse click coordinates are based on the client area of the form.

l) *Y*

m) *ToString*

|           |         |            |          |              |
|-----------|---------|------------|----------|--------------|
| [C#]      | public  | int        | Y        | {get;}       |
| [C++]     | public: | __property | int      | get_Y();     |
| [VB]      | Public  | ReadOnly   | Property | Y As Integer |
| [JScript] | public  | function   | get Y()  | : int;       |

#### *Description*

Gets the y-coordinate of a mouse click.

The mouse click coordinates are based on the client area of the form.